Lecture Notes in Artificial Intelligence 2744

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Vladimír Mařík Duncan McFarlane
Paul Valckenaers (Eds.)

Holonic and Multi-Agent Systems for Manufacturing

First International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2003
Prague, Czech Republic, September 1-3, 2003
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Vladimír Mařík
Czech Technical University, Faculty of Electrical Engineering
Technicka 2, 16627, Prague 6, Czech Republic
and
Rockwell Automation Research Center Prague
Americká 22, 12000 Prague 2, Czech Republic
E-mail: marik@labe.felk.cvut.cz

Duncan McFarlane
University of Cambridge, Institute for Manufacturing
Mill Lane, Cambridge, CB2 1RX, UK
E-mail: dcm@eng.cam.ac.uk

Paul Valckenaers
Katholieke Universiteit Leuven
Mechanical Engineering Department - P.M.A.
Celestijnenlaan 300 B, 3001 Leuven, Belgium
E-mail: Paul.Valckenaers@mech.kuleuven.ac.be

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): I.2.11, I.2, J.1, D.2, I.6

ISSN 0302-9743
ISBN 3-540-40751-0 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP Berlin GmbH
Printed on acid-free paper SPIN: 10929254 06/3142 5 4 3 2 1 0

Preface

The increasing complexity of manufacturing systems as well as the overall demands for flexible and fault-tolerant control of production processes stimulates (among many others) two key emerging technologies that are already making an important breakthrough in the field of intelligent manufacturing, control, and diagnostics. These two paradigms are:

- the **holonic approach** based on the event-driven control strategy, usually aimed at modular control systems that are directly physically linked with the manufacturing hardware equipment, and
- the multi-agent approach developed in the area of distributed information processing.

The research communities working in both these fields are approaching the problem of intelligent manufacturing from different viewpoints and, until recently, to a certain extent, in an independent way. We can however observe quite a clear convergence of these fields in the last few years: the communities have started to cooperate, joining efforts to solve the painful problems involved in achieving effective industrial practice. We can see convergence in the terminology, standards and methods being applied.

The shift in the focus of holonic research over the last five years is significant: Whereas Phase I of the HMS (Holonic Manufacturing Systems) Consortium project was aimed primarily at low-level real-time control and resulted, for example, in the IEC 61499 standard for real-time function-block oriented holonic control, the proposals for Phase II of this consortium are much more concentrated on agent-based control solutions, including knowledge management methods, simulation tools and standards developed by the multi-agent community within the framework of the FIPA consortium activities. The best evidence of this trend is the title and contents of the recently published book

M.S. Deen (ed.): *Agent-Based Manufacturing –Advances in the Holonic Approach,* Springer-Verlag, Heidelberg, 2003

On the other hand, the multi-agent community realizes to a much wider extent than before that one of the most challenging areas for the application of the agent-based information and decision-making systems is the field of intelligent manufacturing. More and more agent-based systems have started to appear in the manufacturing domain, for simulation, production planning and scheduling goals, and for reconfiguration purposes. In the latter, coalition formation and teamwork planning methods are often applied for this purpose. This trend is strongly supported by the challenging visions of virtual manufacturing and virtual enterprises. The other important emerging challenge is the opportunity to consider each product or semiproduct as an agent/holon – this vision is supported by, for example, the recent results of the AUTO-ID initiative aimed at integrating RFID tagging technology with a global information network.

We should stress that the convergence mentioned above is catalyzed mainly by real industrial needs: a very high flexibility in changing production plans and schedules, real-time fault detection and reconfiguration capabilities, as well as allowing

autonomous subsystems to be easily integrated with various kinds of communication networks are stressed more and more. Satisfying these more complex and demanding requirements needs an exploration of knowledge-intensive solutions as well as the leveraging of the current communication and database technologies. The holonic and multi-agent ideas support trends towards higher degrees of local autonomy connected with larger volumes of knowledge "owned" locally in the control elements, and towards more powerful processors (capable of running real-time, ladder-logic types of code as well as higher-level code written, for example, in C++ or Java in parallel) and Web-enabled devices.

Many topics important for agent-based manufacturing are still in the early stages of their research. This is the case in, for example, semantics and ontologies (it would be desirable to use technologies similar or compatible to those in the Semantic Web area, but this seems to be currently more than difficult), or applying holarchy principles in the agent-oriented solutions.

We hope that the first three HoloMAS workshops held under the DEXA-event umbrella (HoloMAS 2000 in Greenwich, HoloMAS 2001 in Munich and HoloMAS 2002 in Aix-en-Provence) helped to bridge the gap between the holonic and multi-agent communities. The interest of both the camps has been increasing and because we have seen there is a chance to cover the whole scene, we decided to transform the HoloMAS workshop into a conference, the **1st International Conference HoloMAS 2003** within the framework of the DEXA 2003 event. We are very thankful to the DEXA Association (Linz, Austria) for supporting the conference idea from the very beginning and to the EU Centre of Excellence MIRACLE at the Czech Technical University of Prague for general support.

We are very glad to declare that 43 papers were submitted, prepared by the most important research bodies engaged in holonic and agent-based manufacturing world-wide to HoloMAS 2003. The PC chose 29 papers to be presented and included in this volume. They contain the most representative results of the corresponding research inside and outside the HMS consortium and provide an excellent overview of the current situation in this subject's research field.

The HoloMAS 2003 conference created an excellent, highly motivating environment, and helped to integrate the community. We believe that it contributed to a better clarification of the goals and to a more efficient coordination of the research in this subject field. The conference also aimed to serve as a display window of holonic and agent-based manufacturing research, offering information about the state-of-the-art to specialists in neighbouring, knowledge-processing research fields covered by the DEXA multi-conference event.

Prague, Cambridge, Leuven
June 2003

Vladimir Mařík
Duncan McFarlane
Paul Valckenaers

HoloMAS 2003

1st International Conference on Industrial Applications of Holonic and Multi-agent Systems, HoloMAS 2003

Applications of Holonic and Multi-agent Systems

Prague, Czech Republic, September 1–3, 2003

Program Co-chairs

Vladimír MAŘÍK	Czech Technical University in Prague, and Rockwell Automation, Czech Republic
Duncan McFARLANE	University of Cambridge, UK
Paul VALCKENAERS	Katholieke Universiteit, Leuven, Belgium

Program Committee

Hamideh AFSARMANESH	University of Amsterdam, The Netherlands
Franz AUINGER	Profactor, Austria
Jose BARATA	Universidade Nova de Lisboa, Portugal
Jeff BRADSHAW	University of West Florida, USA
Robert W. BRENNAN	University of Calgary, Canada
Sven BRUECKNER	ALTARUM, USA
Luis M. CAMARINHA-MATOS	Universidade Nova de Lisboa, Portugal
Eduardo CASTELLANO	IKERLAN, Spain
Armando COLOMBO	Schneider Group, France
Misbah DEEN	University of Keele, UK
Klaus FISCHER	DFKI GmbH, Germany
Martyn FLETCHER	Agent Oriented Software Ltd., UK
William GRUVER	Simon Fraser University, Canada
Kenneth HALL	Rockwell Automation, USA
Matthias KLUSCH	DFKI GmbH, Germany
Dilip B. KOTAK	National Research Council of Canada, Canada
Jiří LAŽANSKÝ	Czech Technical University, Czech Republic
Peter LUH	University of Connecticut, USA
Francisco MATURANA	Rockwell Automation, USA
Gerard MOREL	CRAN, France
Jörg MÜLLER	Siemens AG, Germany
Douglas NORRIE	University of Calgary, Canada
Michal PĚCHOUČEK	Czech Technical University, Czech Republic
Gregory PROVAN	Rockwell Scientific Company, USA
Leonid SHEREMETOV	Mexican Oil Institute, Mexico
Alexander SMIRNOV	SPIIRAS, Russia

Table of Contents

Holonic Manufacturing Systems – Device Control

Foundations/Platforms

Scheduling and Resource Allocation

Simulation and Integration

Multi-agent Systems

Applications

Holonic Manufacturing Systems: Phase II

William A. Gruver[1], Dilip B. Kotak[2], Edwin H. van Leeuwen[3], and Douglas Norrie[4]

[1] School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6 Canada
gruver@cs.sfu.ca

[2] Systems Integration, Testing & Evaluation, NRC Innovation Centre, Vancouver, V6T 1W5 Canada
dilip.kotak@nrc-cnrc.gc.ca

[3] Discovery Technologies, BHP Billiton, GPO Box 86A, Melbourne Victoria 3000, Australia
edwin.h.vanleeuwen@bhpbilliton.com

[4] Dept. of Mechanical & Manufacturing Engineering, University of Calgary, Calgary Alberta T2N 1N4 Canada
norrie@enme.ucalgary.ca

Abstract. The Holonic Manufacturing Systems (HMS) Project is an international industrially driven project addressing systemization and standardization, research, pre-competitive development, deployment and support of architectures and technologies for open, distributed, intelligent, autonomous and co-operating systems on a global basis. During its ten-year program, the HMS Project has developed specifications of holonic architectures, a computer-aided environment for the encapsulation, reuse and integration of holonic systems technologies, and libraries of demonstrated, reusable technologies and tools for the construction of holonic manufacturing systems. These next-generation holonic systems are providing the flexibility, agility and robustness necessary for the rapid delivery of custom manufactured products in competitive global markets. By building on the results of Phase 1 of the HMS Project that emphasized the development of generic technologies and their demonstration in specific application areas, Phase 2 will achieve their integration and application at three levels of the manufacturing and supply chain enterprise: Holonic Control Devices, consisting of the technical manufacturing equipment; Holonic Production Sites and Physical Equipment, comprising the manufacturing work cells; Holonic Planning and Execution Systems, providing scheduling and control of the holonic manufacturing system and a virtual test bed environment for holonic system implementations; and Holonic Man-Machine and Emulation Systems, a virtual test-bed environment for holonic system implementations.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 1–14, 2003.

1 Background

Holonic manufacturing systems (HMS) are based on highly decentralized manufacturing control systems, built from a modular mix of semi-standardized, autonomous, cooperative, and intelligent elements. The functional and structural concepts of HMS are derived from the general concepts of life sciences, such as, biology, psychology and social sciences. The new paradigm combines natural concepts from dynamic hierarchical systems with integration of autonomous elements in distributed systems. HMS is based on the pioneering work of Arthur Koestler [1] in the late 1960s on the modeling of biological and social systems as systems consisting of self-contained elements and capable of functioning as autonomous entities in a cooperative environment. The term holon, introduced by Koestler to describe such an element, is a fusion of the Greek word 'holos' meaning whole and the suffix 'on' denoting a particle. Other approaches to distributed control include heterarchical control [2,3], subsumption architectures [4], and bionic architectures [5]. Each of these approaches embodies desirable system behaviors and may be functionally implemented in a holonic manufacturing system.

The Holonic Manufacturing Systems (HMS) Project is an international, industrially driven project addressing systemization and standardization, research, pre-competitive development, deployment and support of HMS architectures and technologies for open, distributed, intelligent, autonomous and co-operating systems through a global partnership. It is a major project of the Intelligent Manufacturing Systems (IMS) Program [6,7]. The HMS Consortium includes partners from Australia, Canada, Japan, European Union, and USA, comprising large and small industrial users, vendors, universities, and research institutes. Its principal goal is the advancement of the state of the art in discrete, continuous, and batch manufacturing through the integration of highly flexible, reusable and modular manufacturing.

The HMS Project is developing specifications of holonic architectures, an environment for the encapsulation, reuse and integration of holonic system technologies, and libraries of demonstrated, reusable technologies and tools for the construction of holonic manufacturing systems. Beginning in 1992 with a feasibility study known as the Test Case, the partners in the HMS project examined the requirements of 21st century manufacturing systems. To satisfy these requirements, it was determined that an approach based on holonic system technologies was to be developed. In Phase 1 of the HMS Project during 1995-2000, research was conducted in three major areas: (1) generic technologies, (2) benchmarking and test beds in specific application domains, and (3) organizational activities of project management, technology transfer and dissemination, and exploitation. Phase 1 was organized under the following seven work packages:

- *Systems Architecture and Engineering*
- *Systems Operation*
- *Holonic Resource Management*
- *Holonic Manufacturing Unit*

- *Holonic Fixturing*
- *Holonic Handling Systems*
- *Holomobiles*

Not only did the results of Phase 1 demonstrate the feasibility of using holonic technologies for the control of devices and manufacturing work cells, it also indicated its potential to control any distributed system including manufacturing and supply chain management. This expanded scope is reflected in the organization of work packages for Phase 2. Whereas Phase 1 of the HMS Project emphasized the development of generic technologies and their demonstration in specific application areas as documented in proprietary reports and software, and publications in the open literature [8-28], the focus of Phase 2 is the integration and application of HMS technologies at the following four levels of the manufacturing and supply chain enterprise:

- *Holonic Control Devices* targeting the technical manufacturing equipment level;
- *Holonic Production Sites and Physical Equipment* treating the manufacturing work cell level consisting of a number of devices;
- *Holonic Planning and Execution Systems* dealing with scheduling and control of holonic manufacturing systems at the factory and supply chain levels and providing a virtual environment to create test bed for holonic system implantations.

2 Holonic Manufacturing Systems

The purpose of a holonic manufacturing system is the production of material goods by applying value-adding transformations to materials, i.e., perform manufacturing tasks such as discrete-parts fabrication and assembly, chemical reactions and purifications, and packaging of solid and liquid products.

2.1 Device Management

Physical holons are thus indispensable parts of all HMS. The physical holons shown in Fig. 1 consist of production equipment capable of performing manufacturing tasks and one or more associated holonic control devices with the following functions:

- *Inter-holon communication* to negotiate and coordinate the execution of processing plans (sequences of manufacturing tasks) and recovery from abnormal operations;
- *Real-time control functions* that implement and monitor the required sequences of operations as well as detecting and diagnosing malfunctions;
- *Physical interfaces* between the control functions and the sensors and actuators of the physical processing equipment.

These functions can reside in layers of a holonic control device with appropriate inter-layer communication (Fig. 2).

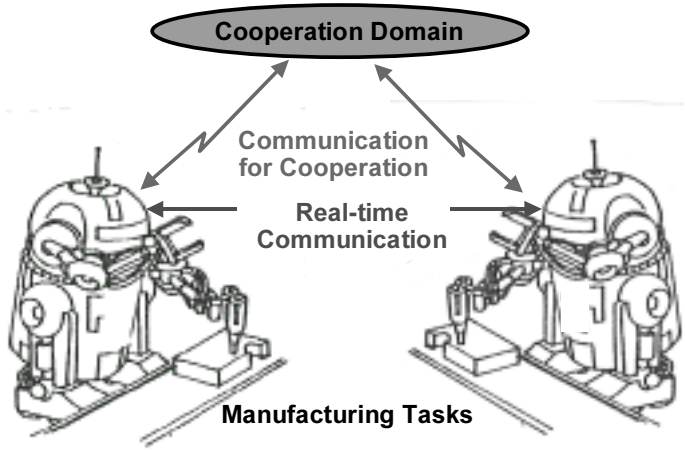


Fig. 1. Physical holons

The nature of a holonic manufacturing system poses the following new requirements:

- Highly diverse physical equipment requires that holonic control devices be implemented on a wide variety of interoperable control platforms in heterogeneous, distributed control systems;
- Rapidly reconfigurable physical equipment requires that holonic control devices also be rapidly reconfigurable, preferably automatically by software agents; i.e., it should be mostly self-reconfiguring with minimum delays for human intervention;
- The need for human integration requires that appropriate interfaces to facilitate full use of human skills are integrated at all functional levels, not separately designed add-ons.

To achieve these high level goals, basic requirements must be fulfilled, including:

- Control software encapsulation, reuse and portability;
- Dynamic reconfigurability, platform and communication stack independence of control applications and human interfaces.

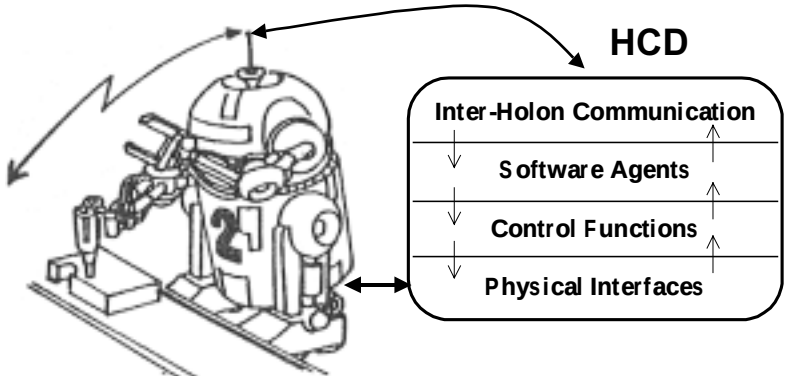


Fig. 2. Holonic control device (HCD)

2.2 Work Cell and Factory Management

Most factories still are islands of technologies not properly integrated. There is a proliferation of technologies in the manufacturing industry: MRP/ERP, CAD/CAM/CAPP, maintenance management systems, machine level control and SCADA systems. However, most of them operate within their own domains, some of them optimize specific operations, but none address the issue of factory wide optimization and coordination.

One of the difficulties in achieving factory wide coordination and control is that decisions are made at different times, and in different places by people with different responsibilities, perspectives, and priorities. Frequently, the decisions made by one individual may undermine and negate the effect of decisions made by someone else. The result is that most factories are managed by operational personnel running from crisis to crisis, with no opportunity for even local optimization, let alone global optimization.

Holonic manufacturing principles rely upon control and coordination by cooperation through negotiations (Fig. 3). In a crisis when a centralized control or coordination is not possible, holonic systems still permit autonomous operations. Thus, the focus of the HMS Project (Phase 2) is to create integrated holonic manufacturing systems -- using virtual manufacturing test beds -- and demonstrate the capabilities of the technology.

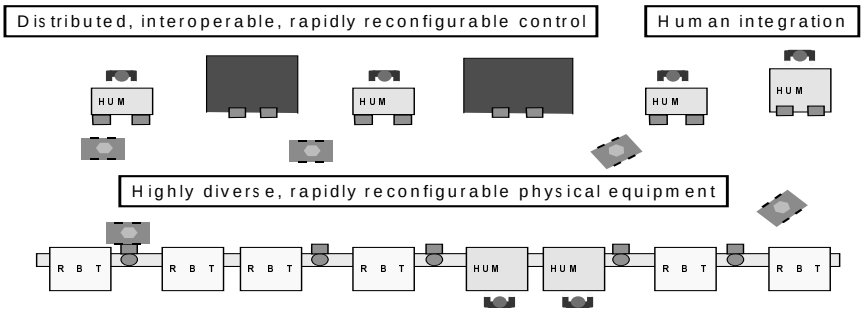


Fig. 3. Holonic factory

2.3 Supply Chain Management

Supply chain management is an optimization technology in which intelligent algorithms supply the backbone for advanced planning and scheduling applications designed both as a custom solution and as a growing number of commercial packages. SCM is primarily involved in making key corporate data accessible to improve productivity. This function is achieved by the optimization of functions throughout the supply chain, ranging from procurement to manufacturing, distribution, and sales. Planning answers the question "what should I do?" whereas scheduling is a process that treats "how should I do it?" Manufacturing scheduling synchronizes the supply of materials with workflow and resources to optimize production. Scheduling determines the optimal grouping and sequencing of orders on the shop floor based on detailed product attributes, production line capacities, and material flow. Once the finished product leaves the plant, optimization can minimize transportation costs by consolidating shipments into full truckloads or by pairing the least costly shipping mode with timing that still respects the customer due date.

A real-time supply chain execution system provides rapid feedback on customer orders and available-to-promise or capable-to-promise capabilities. Holonic systems and optimization provide the tools to guarantee effective reaction to customer demand.

3 Project Structure

The objectives of the Holonic Control Devices (HCD) work package are as follows:

- Develop a set of specifications for HCD and their associated interfaces, in a form suitable for international standardization, and submit these specifications to appropriate standardization bodies such as IEC, ISO and FIPA;

- Develop, test and demonstrate prototype implementations of HCD that complying to these specifications in a number of different applications, to ensure that these specifications are correct, complete, implementable, and meet the practical needs for implementation of physical holons.

3.1 Holonic Control Devices

The scope of this research is limited to the HCD associated with physical holons occupying the lowest layer of a manufacturing holarchy (Fig. 4). This includes those features necessary to enable the physical holons to enter into negotiations about the performance of manufacturing tasks and to mutually coordinate the execution of those tasks. Such features include:

- The ability to locate, join, leave and participate in cooperation domains for the performance of manufacturing tasks;
- The ability to reason about manufacturing tasks and to acquire and share knowledge related to such reasoning;
- The ability to issue appropriate IEC 61499 management commands to dynamically modify existing applications to perform new tasks or to recover from abnormal events.

The results of this work package will enable the global deployment of HMS technologies to the factory floor and it will provide answers to the following questions:

- What are the benefits of applying HMS technologies to physical factory-floor equipment?
- Which technologies are most beneficial for specific types of machines and processes?

3.2 Holonic Production Sites and Physical Equipment

The objectives of the Holonic Production Sites and Physical Equipment (HPS) work package are as follows:

- To deploy, evaluate and further improve available HMS technologies developed by the HMS consortium, including control architectures, contract net protocols, negotiation principles, cooperation domain principles, control strategies, and communication infrastructures;
- To systematize and develop application guidelines and configuration rules for the construction and operation of holonic production sites and physical equipment in relation to shop floor layout, device redundancy, human integration, task allocation and its relationship to resulting system behavior, throughput and productivity;
- To implement and operate physical holonic production sites and physical equipment on an international level, by integrating HMS technologies in mechanical hardware, and computer hardware and software;

- To demonstrate and prove the technical and economical benefits of the implemented holonic production sites and physical equipment under industrial conditions, thereby motivating end-users to apply HMS technologies and to enable control system suppliers to engage in their exploitation and marketing.

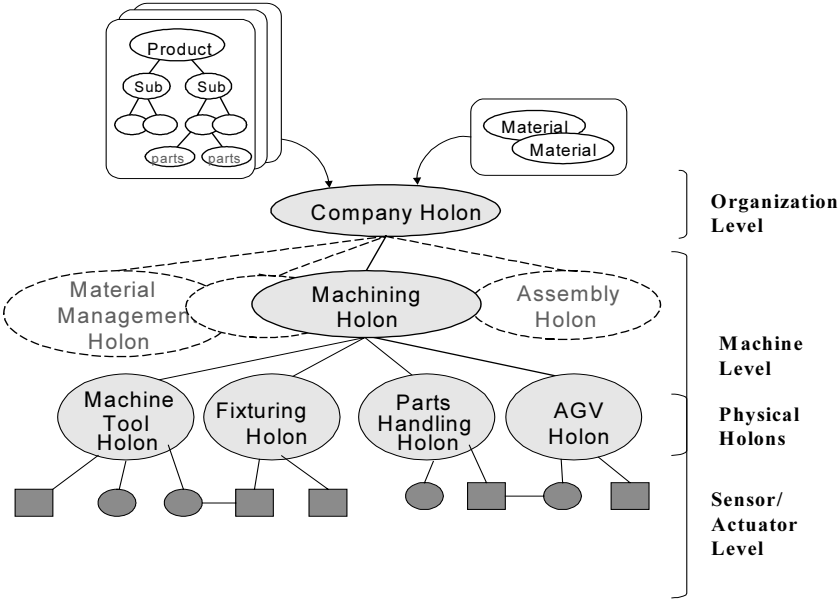


Fig. 4. Physical holons in a manufacturing holarchy

The HPS work package focuses on the implementation and operational aspects of holonic production sites at the production line, shop floor and work cell levels. Emphasis is given to the implementation and operational aspects of shop floor layout, control architecture, human integration, resource allocation, operation and system behavior. Activities of this work package cover deployment and the development of new concepts, and the systematization of existing methodologies. The HPS work package will improve the understanding, methods and principles for implementation, deployment and operation of HMS technologies at the shop floor level. These results will enable the control system suppliers to amend products, tools and services and finally the end users to make optimal usage of the potentials of HMS technologies.

This work package will provide answers to the following questions:

- How to design holonic production sites and related physical equipment. What are the holons? How much redundancy of production capacity is appropriate? Which factory layout is most suitable?

- Which types of control architecture and control systems are required and are supportive?
- How to implement and integrate the HPS;
- How to operate and monitor the HPS.

3.3 Holonic Planning and Execution Systems

The objectives of the Holonic Planning and Execution Systems (HOPES) work package are as follows:

- To develop adaptive algorithms and methods to support decentralized decision-making processes;
- To realize the automation of these algorithms and methods to enable information-processing systems for self-organization;
- To increase the reactivity to changes in the manufacturing environment by fast autonomous decisions;
- To develop, implement and test methodologies for decentralized, agent-based scheduling and an integrated scheduling and process planning;
- To develop, implement and test a scheduler for business control based on Lagrangian relaxation;
- To develop, implement and test concepts and models for agent-based business networking, supply chain management;
- To develop, implement, and test functions to develop and operate holonic production execution systems.

This work package focuses on the transfer of methods and principles of Holonic Manufacturing Systems to higher levels of company planning and production execution within single and crossover several companies within a supply chain. It comprises the development, implementation, emulation and test of a holonic production site on production line, shop floor and work cell level. Emphasis is placed on:

- Multi-user man-machine interface environment to permit decision makers and other users within a manufacturing or supply chain enterprise to effectively interact with the system at all levels;
- Design and operation system that enable a virtual manufacturing and supply chain environment;
- Information system integration that permits the capture and fusion of information and knowledge coming from the physical, manufacturing, and business systems.

The research conducted in the HOPES work package is intended to demonstrate the technical and economic advantages of applying HMS technologies in industrial enterprise in term of increased reliability, flexibility and productivity, transfer the understanding, methods and principles of HMS to higher levels of company planning and production execution within single and crossover several companies within a supply chain. It will also support the application and further development of the internationally accepted holonic method, concepts, tools, and standards as jointly defined by the HMS consortium. The results of this package will enlarge the field of

application of HMS technologies and enable the system integrators to amend software products, tools and services and finally the end users to make optimal usage of the benefits of HMS technologies.

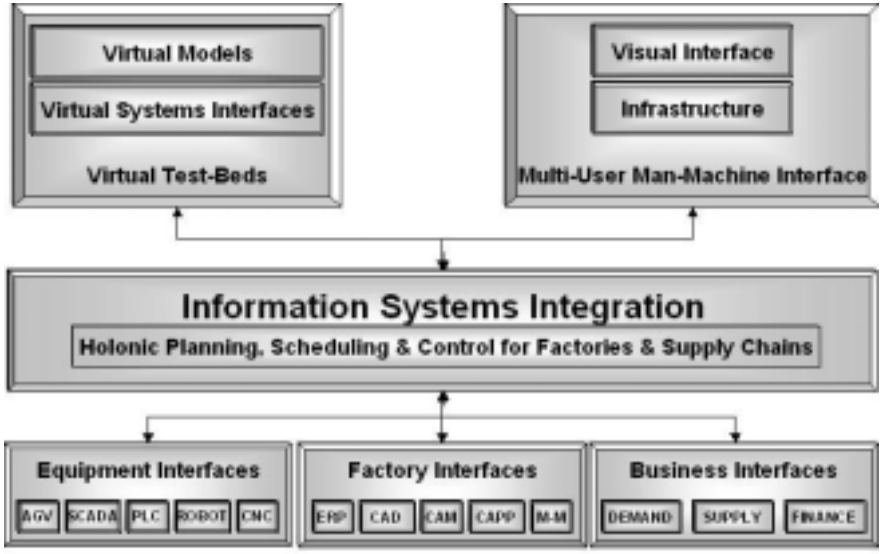


Fig. 5. Structure of the HOPES Work Package

The HOPES work package will provide answers to the following questions:

- What is the benefit of applying HMS technologies higher levels of company planning and production execution?
- Which technologies are most beneficial for specific company requirements?
- How can these technologies be implemented in daily business?

The structure and major elements of the HOPES work package are shown in Fig. 5.

4 Holonic Manufacturing Systems Architecture

A holonic manufacturing systems architecture integrates all levels (devices, work cells, factories and supply chains) to provide a highly distributed system that supports a modern global enterprise. A widely used approach for modeling holonic manufacturing systems is PROSA (Product, Resources, Orders and Staff Architecture) [27], consisting of the following elements:

- Product Holon –comprising of the physical products being produced and the human and computing support to initiate and monitor the activity to produce it;

- Resource Holon –comprising one or more physical processes or transportation resources, its control systems and any necessary human based operations;
- Order Holon –representing the requirements of a particular order including information such as product qualities, due dates, costs and priorities;
- Staff Holon – an optional support element providing coordination between holons and ensuring that global goals are achieved.

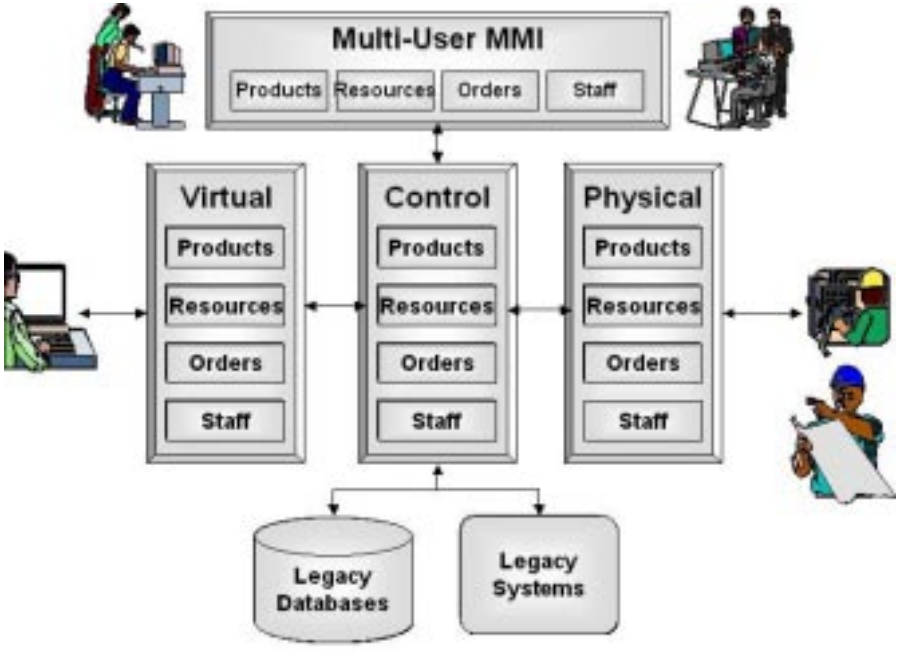


Fig. 6. Holonic manufacturing systems architecture

The physical world in Fig. 6 may represent devices, work cells, factories and supply chains. The legacy databases represent commonly used systems such as CAD, CAM, CAE, CAPP, MRP/ERP, Maintenance Management, Accounting and Finance, Marketing and Sales. Since these systems are typically distributed throughout the organization, the multi-user man-machine interface provides context sensitive information to decision makers when required. A virtual holonic environment provides a risk free means to assess alternatives. It also provides a platform for assessment of holonic coordination and control strategies prior to their implementation in physical systems. A virtual environment can be used to optimize the design and operations of manufacturing or supply chain enterprises. The holonic control environment contains the intelligence that can control virtual holons, physical holons or provide a hybrid environment mixing the physical and virtual environments to test holonic control strategies. One of the keys to use of this architecture is its implementation using network-based communications. By this means holonic system architectures provide the capability to integrate a modern manufacturing enterprise and offer a migration path for legacy systems.

5 Conclusions

The Holonic Manufacturing Systems Program began with an abstract concept of holons. Phase 1 provided a foundation for the development of generic technologies and their application. The results of Phase 2 are establishing the feasibility of using holonic systems for the control of devices and manufacturing work cells. It is demonstrating the potential to control many types of distributed systems ranging from physical equipment to supply chains. This expanded scope is reflected in the Phase 2 work packages specifically aimed at the integrated application of holonic technologies using the same fundamental architecture for device, work cell, factory and supply chain levels. Holonic system architectures have attracted interest by researchers throughout the world [29-33]. HMS provide modern enterprises with the capability to integrate modern manufacturing and supply chains, and offer a graceful migration path from current legacy systems to next generation fully distributed manufacturing systems. The final outcome of the research program will be software tools, software standards, and architectural approaches for holonic systems, proven through industrially driven test-beds. It is expected that these will form the basis of future software technologies through commercialization of the research.

The research takes advantage of recent advances in distributed software, hardware, and control based upon holons that collaborate through negotiations to dynamically respond to perturbations in the production environment. The resulting technology is a new paradigm for distributed manufacturing that is more agile, flexible, robust, and scalable when compared to rigid traditional manufacturing systems. The impact of this new technology is shorter lead times, reduced in-process inventories, and improved profitability. The approaches used in holonic manufacturing systems are extendable to other environments including coordination of supply chain logistics and infrastructures for transportation and energy. Because human decision makers also can be modeled by holons, they could be used in fully automated systems and those that require human integration. Finally, due to the agile and robust characteristics of holonic manufacturing systems, they are particularly suited to safety-critical applications.

Acknowledgements. The authors acknowledge the contributions to this research from the partners of the Holonic Manufacturing Systems Consortium. Particular thanks go to Jim Christensen, Chris Schaeffer, Masako Sudo, Shinsuke Tamura, and Michael Zwick for their leadership of the workpackages.

References

1. Koestler, A., *The Ghost in the Machine*, Hutchinson & Co, 1967; Arkana Books, 1989.
2. Gruver, W. A., and J. Boudreaux, *Intelligent Manufacturing: Programming Environments for CIM*, Springer-Verlag, London, 1993.
3. Duffie, N. A., and V. V. Prabhu, "Real-Time Distributed Scheduling of Heterarchical Manufacturing Systems," *J. Manufacturing Systems*, 13 (2), 1994.

4. Brooks, R. A., "A Robust Layered Control System for a Mobile Robot," *IEEE J. of Robotics and Automation*, 2 (1), 1986.
5. Okino, N., "Bionic Manufacturing Systems," *Flexible Manufacturing Systems, Past-Present-Future*, J. Peklenik (ed), Faculty of Engineering, Ljubljana, 1993, pp 73–95.
6. van Leeuwen, E. H. and D. Norrie, "Intelligent Manufacturing: Holons and Holarchies," *Manufacturing Engineer*, 76 (2), April 1997, pp 86–88.
7. Parker, M., "The Intelligent Manufacturing Systems (IMS) Initiative," *Manufacturing Engineering*, February 1997.
8. Hayashi, H., "The IMS International Collaborative Program," *Proc. of 24th International Symposium on Industrial Robots*, 1993.
9. Christensen, J. H., "Holonic Manufacturing Systems, Initial Architecture and Standards Directions," *Proc. of 1st European Conf. on Holonic Manufacturing Systems*, Hannover, Germany, 1994.
10. Winkler, M. and M. Mey, "Holonic Manufacturing Systems," *European Production Engineering*, Oct 1994.
11. Bongaerts, L., "*Integration of Scheduling and Control in Holonic Manufacturing Systems*," PhD thesis, Katholieke Universiteit, Leuven, Belgium, 1998.
12. Bussmann, S., "An Agent-Oriented Architecture for Holonic Manufacturing Control," *Proc. of IMS '98*, Lausanne, 1998.
13. Bussmann, S., and D. McFarlane, "Rationales for Holonic Control Systems," *Proc. of IMS '99*, Leuven, Belgium, 1999.
14. Deen, S. M., "Cooperation Issues in Holonic Manufacturing Systems," H. Yoshikawa and J. Goossenaerts (eds.), *Information Infrastructure Systems for Manufacturing*, Elsevier Science BV, 1993, pp 401–412.
15. Gayed, N., Jarvis, D., and J. Jarvis, "A Strategy for the Migration of Existing Manufacturing Systems to Holonic Systems," *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, San Diego, CA, 1998, pp 319–324.
16. Gou, L., Luh, P., and Y. Kyoya, "Holonic Manufacturing Scheduling: Architecture, Cooperation, Mechanism, and Implementation," *Computers in Industry*, 37 (3), 1998, pp 213–231.
17. Heikkila, T., M. Jarviluoma, and T. S. Juntunen, "Holonic Control for Manufacturing Systems, Design of a Manufacturing Robot Cell," *Integrated Computer Aided Engineering*, 4, 1967, pp 202–218.
18. D. Kotak, S. Bardi, W. A. Gruver, and K. Zohrevand, "Comparison of Hierarchical and Holonic Shop Floor Control using a Virtual Manufacturing Environment," *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, Nashville, TN, October 2000.
19. S. Liu, W. A. Gruver, and D. B. Kotak, "Holonic Coordination and Control of an Automated Guided Vehicle System," J. of Computer Aided Engineering, Vol. 9, No. 3, 2002, pp 235–250.
20. McFarlane, D., B. Marett, G. Elsley and J. Jarvis, "Application of Holonic Methodologies to Problem Diagnosis in a Steel Rod Mill," *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, Vancouver, Canada, October 1995.
21. McFarlane, D. C., "Holonic Manufacturing Systems in Continuous Processing, Concepts and Control Requirements," *Proc. of ASI '95*, Portugal, 1995.
22. McFarlane, D. and S. Bussman, "Developments in Holonic Production Planning and Control," *Int. J of Production Planning and Control*, 2000.
23. Rannanjarvi, L. and T. Heikkila, "Software Development for Holonic Manufacturing Systems," *Computers in Industry*, 37 (3), 1998, pp 233–253.
24. Tharumarajah, A. and A. Wells, "Behaviour-Based Approach to Scheduling in Distributed Manufacturing Systems," *Integrated Computer-Aided Engineering*, 1996.
25. van Brussel, H. W., J. Valckenaers, P. Bongaerts, L. Peeters, "Reference Architecture for Holonic Manufacturing Systems," *Computers in Industry*, 37 (3), 1998, pp 255–274.

26. van Brussel, H., P. Valckenaers, F. Bonneville, “Programming, Scheduling and Control of Flexible Assembly Systems,” *Manufacturing Systems*, 23 (1), 1994.
27. Wyns, J., *Reference Architecture for Holonic Manufacturing Systems*, PhD Thesis, Katholieke Universiteit, Leuven, Belgium, 1999.
28. Zhang, T., K. Stanley, M. H. Smith, and W. A. Gruver, “Multi-Agent Techniques in Holonic Manufacturing Systems,” *Proc. of 2nd Int. Symposium on Intelligent Manufacturing Systems*, Sakarya, Turkey, August 1998.
29. Kusumi, N., K. Hirasawa, and M. Obayashi, “Holonic Control System Based on a Universal Learning Network,” *Electrical Engineering in Japan*, 124 (4), 1998, pp 64–72.
30. Langer, G., *A Methodology and Architecture for Holonic Multi-Cell Control Systems*, PhD Thesis, Technical University of Denmark, Lyngby, Denmark, 1999.
31. Marcus, A., T. Vancza, and L. Monostori, “A Market Approach to Holonic Manufacturing,” *Annals of CIRP*, 45 (1), 1996, pp 433–436.
32. Matthews, J., “Organisational Foundations of Intelligent Manufacturing Systems - The Holonic Viewpoint,” *Computer Integrated Manufacturing Systems*, 8 (4), 1995, pp 237–243.
33. Tharumarajah, A., A. Wells, and L. Nemes, “Comparison of the Bionic, Fractal and Holonic Manufacturing Systems Concepts” *Int. J. of Computer Integrated Manufacturing*, 9 (3), 1996, 217–226.

A Mechanism for Ensuring Safe Behaviors of Holonic Manufacturing Systems

Shinsuke Tamura, Naoya Nishi, and Tatsuro Yanase

Fucalty of Engineering , Fukui University 3-9-1, Bunkyo, Fukui 910-8507, Japan
tamura@fuis.fuis.fukui-u.ac.jp,
{h4nisi,yanase}@radio.fuis.fukui-u.ac.jp

Abstract. This paper describes a mechanism to ensure safe behaviors of Holonic Manufacturing Systems (HMSs). The change from low-variety high-volume to high-variety low-volume production requires highly flexible and adaptive manufacturing systems. HMSs, in which decisions are made through cooperation among holons (autonomous and cooperative manufacturing entities), fulfill this requirement by exploiting full abilities of individual elements while eliminating various bottlenecks that exist in conventional systems. However, highly adaptive features induce non-deterministic behaviors of systems, and this makes it difficult to adopt HMSs as primary bases of manufacturing systems. In order to apply HMSs to large and complicated applications, mechanisms that make HMS behaviors more predictable are essential. An HMS safety ensuring mechanism proposed here is one of the attempts to make HMSs more predictable.

1 Introduction

Current manufacturing requires highly flexible and adaptive manufacturing systems in accordance with the change from low-variety high-volume to high-variety low-volume production. Manufacturing systems must be endowed with the capabilities of incremental development and dynamic reconfiguration in order to adapt to the environment in which manufacturing requirements, available technologies and social constraints are changing rapidly. Decentralization is one of the most promising solutions to meet this requirement, and various efforts have been made to decentralize management and control of manufacturing. Holonic Manufacturing System (HMS) architecture emerged in the course of this decentralization [1]. An HMS consists of holons, autonomous and cooperative manufacturing elements, and behaviors of HMSs are decided thorough cooperation among holons. Therefore, system bottlenecks regarding performance (e.g. response time), reliability, extendibility and adaptability, exist in conventional centralized management systems and restrict abilities of individual elements, can be eliminated. Namely in HMSs, decisions are made through cooperation among relevant holons, thus the computational loads required for decisions are distributed to these holons, i.e. are not concentrated on the central manager. Regarding the reliability, autonomous behaviors of individual holons avoid system-

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 15–24, 2003.

wide failures. Individual holons can execute their functions despite malfunctions in any other elements. Also the local and loose connections among holons reduce the number of holons to be modified in the case of system extension. Moreover, functions of holons can be utilized effectively in various conditions, because individual holons that decide their behaviors have comprehensive knowledge of their abilities. In these ways HMSs enhance system flexibility and adaptability at the highest level through effective use of elements while eliminating various kinds of system bottlenecks.

Many HMS application systems have been developed already [2]-[4]. These application systems range from high-level management systems, such as manufacturing scheduling and machining/assembly process planning, to low-level control systems, such as discrete and continuous process control, path planning for transfer machines and robot motion planning. Although many of them are experimental ones, substantial advantages are reported concerning system reconfiguration and incremental system development. However HMSs still have drawbacks regarding use in more complicated applications, and therefore, current HMS applications are limited to small or simple ones, despite the advantages of HMSs for developing large and complex systems. The reasons are, firstly, there is not a powerful cooperation algorithm yet, and secondly, cooperative behavior of elements is less predictable than that managed by central mechanisms. Especially, this unpredictability becomes serious problems for manufacturing systems that require highly safe behaviors. To make HMS architecture more practical for the lager and more complicated manufacturing systems, the following issues must be addressed.

- Reconfigurable cooperation mechanisms
  Despite many years of research, currently available cooperation mechanisms are not powerful enough to realize complicated functions by a single mechanism. Even simple functions require various mechanisms in manufacturing environments where requirements, technologies, etc. change rapidly. To apply HMS architectures as the primary bases of manufacturing systems, incremental development and dynamic reconfiguration capabilities of cooperation mechanisms themselves are essential in addition to those of application mechanisms. In other words, cooperation mechanisms to achieve complicated functions can be developed only by continually improving and extending existing mechanisms.
- Easy behavior prediction mechanisms
  Behaviors of real-time control systems are required to be predictable. They can be operated only after confirming the correctness of their behaviors, because real time control systems assume only limited manual interruption of their behaviors. Therefore, system behavior is carefully evaluated at the design stage of the system or its operations. Usually most behaviors of the system can be confirmed through designers' insight, and models for computer simulation, which are considered to be the most reliable way of predicting system behaviors, are developed for only critical behaviors. The reasons are that simulation model development requires a great effort, and also many behaviors are simple and can be manually predicted. However, behaviors of HMSs are not easy to predict manually even they are simple, since an HMS behaves non-deterministically based on cooperation among holons. Therefore,

simulation models are essential, and they must be developed with the minimum cost in order to make HMSs practical. Moreover, these models must change their configurations continually in accordance with changes in HMSs themselves.

- Safe behavior ensuring mechanisms

Non-deterministic features are a serious drawback of HMSs also from the viewpoint of safety. Safe behavior of a system is ensured through examining various possibilities, but usually it is impossible to enumerate all the possible behaviors. Actually, even trivial safety conditions are overlooked at the design stage frequently, and cause serious problems. In HMSs where combinations of individual holons generate an enormous number of behavior patterns, the possibility that safety conditions are overlooked is much higher than in conventional systems. To apply the HMS architecture extensively, mechanisms that ensure safe behaviors of systems are essential. Simulation models are effective only to confirm the correctness of given behaviors. Formal specification methods, frequently used for designing highly reliable systems, are also effective only for detecting contradictions of given safety conditions. They are not effective for finding missing safety conditions.

In the following sections, a mechanism to address the last issue, i.e. to ensure safe HMS behaviors is proposed. Regarding the first issue, in HMS-shell (an HMS development and operation environment developed in the HMS consortium) [5], cooperation mechanisms are decomposed into finer modules so that various kinds of cooperation mechanisms can be configured by combining different modules. Also in HMS-shell, holons have self-simulation capabilities to constitute self-simulators in order to make HMS behaviors predictable [6]. Here, self-simulators can be used for both actual operations and simulations by simply switching their running modes. Thus, the development processes of simulators are completely included in HMS development processes themselves, and designers can construct simulators and modify them continuously along with system improvements as pure design processes, not as additional processes.

2 Ensuring Safe Behaviors of HMSs

To maintain safe joint activities of autonomous elements is difficult, even if behaviors of individual elements are simple. Various kinds of spatial, temporal and other conditions concerning behaviors of elements must be satisfied to maintain their safety. For example, the sizes of workpieces loaded on machines located at mutually adjacent places cannot exceed certain limits to avoid their collision, a certain interval is necessary between the starting times of tasks A and B belonging to different work-flows, when their succeeding tasks require the same resource and must be started within predefined periods after finishing A and B, respectively. Also a task must use critical data while disabling other tasks from accessing them in order to maintain data consistency. These conditions are difficult to satisfy because they vary with time, and holons are developed without knowledge of these conditions, i.e. individual holons are developed for different purposes at separate places and have different structures. In order not to undermine the advantages of HMSs, mechanisms to ensure HMS

safety must be implemented with the minimum changes in these existing holons, and work well among heterogeneous holons with different structures. Also they must be easy to reconfigure in accordance with changes in HMSs themselves.

A mechanism to ensure safe behaviors of HMSs proposed in this section restrains joint behaviors of holons so that they satisfy predefined conditions, while systematically encapsulating these conditions in order to make the mechanism reconfigurable and accessible by holons developed independently. The mechanism functionally integrates technologies used in data integrity maintenance, concurrency control and access control mechanisms, and structurally it is based on layered (or subsumption) architecture. As shown in Fig. 1, the mechanism consists of two layers, an application layer and a safety layer. Application holons corresponding to machines, materials and product orders, etc. are located in the application layer. They are usually modifications of lagacy holons with heterogeneous structures. Holons in the safety layer (safety holons) are defined corresponding to critical variables or methods of application holons, and responsible for restricting joint behaviors of application holons so that they satisfy conditions regarding those variables or methods. In this configuration, behaviors of application holons are checked by safety holons whenever application holons access critical methods and /or variables, in order to maintain safety of HMSs.

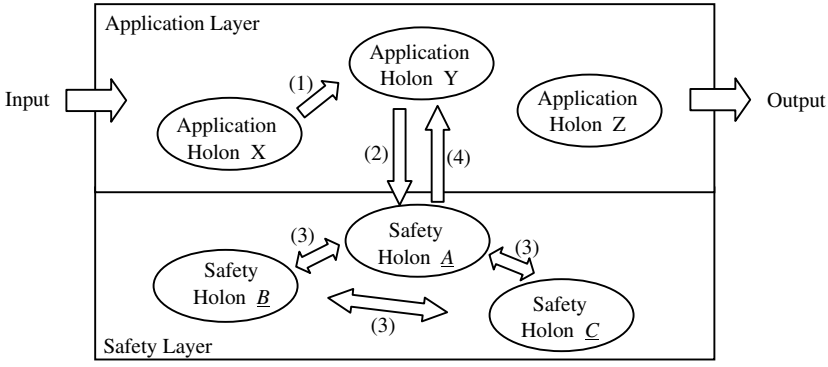


Fig. 1. Architecture to ensure HMS safety

The mechanism works as follows. That is, (1) when an application holon X accesses a variable or a method A of some holon, (2) the variable/method A invokes the safety holon *A* that corresponds to A, and (3) the safety holon *A* invokes safety holons *B* and *C* corresponding to variables or methods B and C, which are relevant to A, respectively. Then safety holons *A*, *B* and *C* cooperatively maintain or check safety conditions, and finally, (4) the results are returned to A to be transferred to the application holon X. Here, safety holons are designed and implemented with their global names independently of application holons, and application holons can access safety holons by designating their global names. The only modification of existing application holons in order to adapt the safety mechanism is the declaration of safety vari-

ables/methods. Namely, when an application holon declares its accessing variables and methods as safety ones with their global identifiers in the source code as shown in Fig. 2, mechanisms for invoking corresponding safety holons that have the names coinciding with the global identifiers are generated automatically, when source codes of application holons are compiled.

```
Safety variable A= A;
Safety method T= T;

/* A and T are a variable and a method relevant to
safety conditions, and A and T are their global
identifiers  */
```

Fig. 2. Safety variable and method declaration

By means of this structure, the following types of safety constraints can be expressed.

1) Value Constraints

These constraints maintain safe behaviors of HMSs as data integrity constraints do for database systems. Namely, behaviors of holons are restricted so that values of critical variables belonging to cooperating holons satisfy the predefined arithmetical and /or logical relations. Unlike usual data integrity constraints, which restrict values of variables in a single database, these constraints restrict values of different holons implemented on heterogeneous platforms.

2) Access Constraints

These are the extensions of constraints used in access control mechanisms, i.e. only entities that show access rights coincide with the predefined ones are allowed to accesses the data/methods as in usual access control mechanisms. These access rights include dimensions of data to be read and/or written such as "kg" and "m^2," objectives of accesses such as "work-piece picking" and "location adjustment," and access and other state histories of accessed and /or accessing holons such as "after invoking the method to wash tools."

3) Protocol Constraints

These constraints force holons to access methods and variables with consistent procedures. For example, accesses that do not comply with defined lock and commit protocols can be prevented. The proposed structure makes it easy to implement and modify application specific access protocols dynamically.

These constraints are extensions of conventional mechanisms to ensure safety that correspond to value checking, access right checking and concurrency control, respectively. The important features of the proposed mechanism are that, firstly, various constraints concerning joint behaviors of holons can be integrated in single safety holons, secondly, holons with different structures designed and developed at different places for different purposes can be easily incorporated into this architecture, i.e.

interoperability, and thirdly, the mechanism can be dynamically reconfigured easily. Explicit separation of safety holons from application holons in the proposed architecture provides advantages in terms of enhancing safety behaviors of manufacturing systems, and their reconfiguration and incremental development capabilities, as follows.

- Safety behaviors
  Conditions for safe behaviors can be extracted from inter-holon perspectives without being disturbed by considerations concerning intrinsic functions of application holons. Also durations for designing safety holons are not limited by that of application holons, i.e. safety constraints can be designed even after the completion of application program development. Without the separation, even simple conditions for safety tend to be overlooked, because of complicated application functions and limited design durations.
- System reconfiguration and incremental development
  Configurations and algorithms of holons in both application and safety layers can be changed independently of the other layers. Application holons are required only to declare their safety variables and methods in order to modify safety conditions. Reconfigurability automatically brings about the easy incremental development of manufacturing systems, i.e. both application and safety holons can be easily added and modified, e.g. different manufacturing systems can be incorporated into the system safely by only adding declarations of safety variables and methods in relevant holons.

3 Prototype System Development

A simple experimental program, a valve controller, has been developed in order to evaluate the proposed architecture. As shown in Fig. 3, the valve controller opens and closes valves attached to pipes according to conditions of tanks that are connected to pipes. Here, one of chemicals S, T or U is supplied to tank-X and tank-Y, and mixing S with U is inhibited. Then, open/close operations of valves must satisfy the following constraints to safely control the whole system, i.e. 1) valve A/B must be closed when valve C/D is closed and the level of tank-X/Y exceeds the maximum limit, 2) valve C/D must be closed when valve D/C is open and tank-X and Y contain chemicals S and U, or U and S, respectively.

It is not difficult to imbed mechanisms to check these constraints into application holons if application holons are developed from scratch and safety constraints do not change during operations. However, usually already existing application holons are reused. Also constraints for safe behaviors are changed during operations: for example, when the system is extended to treat chemical V, which is inhibited to mix with T, valve C must be closed also when valve D is open and tank-X and Y contain chemicals T and V, or V and T, respectively. In these cases, it is not easy to imbed new constraints correctly into application holons. Designers are prone to make mis-

takes in environments where safety constraints are tightly connected to application procedures and are distributed over various application holons.

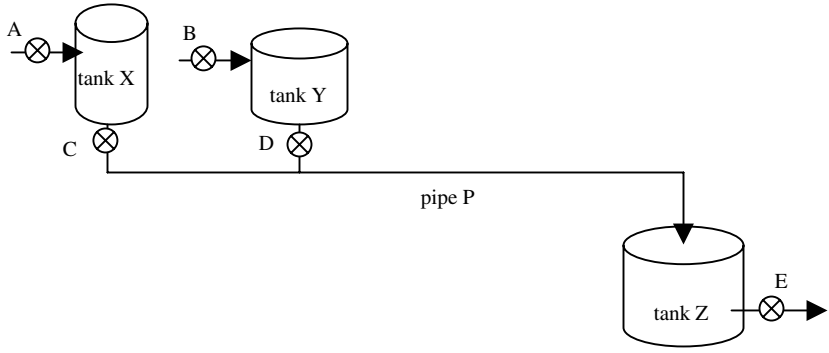


Fig. 3. Valve control system

The experimental system resolved this difficulty by preparing 4 kinds of safety holons as follows.

- Chemical holon: corresponds to an individual chemical and maintains a list of chemicals that are not mixed with it.
- Tank holon: corresponds to an individual tank and maintains names of chemicals that it contains, its maximum level and the names of its output valves.
- Pipe holon: corresponds to an individual pipe and maintains names of chemicals that flow within it.
- Valve holon: corresponds to an individual valve and maintains a list of inputs resources (tanks or pipes) and output resources.

Here an application holon, which manipulates valve A or C, declares variable A or C as safe variables, respectively. When the application holon tries to open valve A, safety holon *A* is invoked automatically, then *A* allows to open the valve only when it is confirmed that tank X will not overflow. This confirmation is carried out by asking the tank level and state of valve C to safety holons *X* and *C*, respectively, which are connected to *A*, as shoun in Fig.4. In the case when the application holon tries to open valve C, the safety holon *C* is invoked to confirm that constraint 2) is satisfied. This is carried out through 2 steps, i.e. *C* asks names of chemicals that pipe P and tank X contain to safety holons *X* and *P*, which are connected to *C*, and then checks that mixing two chemicals is allowed by asking safety holons corresponding chemical names that are answered by *X* and *P*, i.e. safety holon *S*, *U* or *T*.

Figure 5 shows an implementation structure of safety holons appear in Fig.4. Application holons in the application layer are implemented as C^{++} objects. On the other hand, safety holons in the safety layer are implemented as JDPS [7] objects and allocated over distributed computers connected through Ether-net. In the figure, safety

holn *C* is replicated because it relates to the critical variable/method, and the system can avoid the most serious situations even when several computers have broken. However, because the same safety holons are replicated, the system cannot avoid accidents caused by program errors in safety holons.

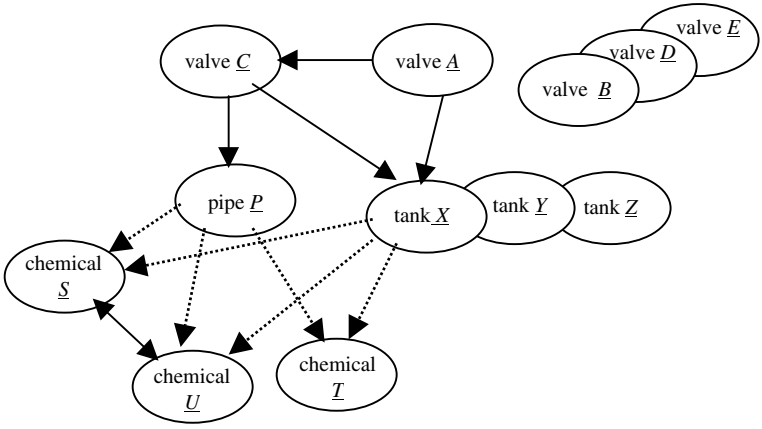


Fig. 4. Safety holons

During the development of the experimental system, it has been shown that critical constraints can be extracted and implemented safely by describing them in terms of root causes such as mixing constraints of chemicals without disturbed by complicated control procedures. Also the experimental system showed the advantages of the proposed mechanism about addition, deletion and modification of safety constraints as follows.

- The system with safety constraint 1) can be extended to the system described above, by adding constraint 2) to safety holon *C*. In this case because safety holon *P* does not maintain a name of the chemical flowing in it, procedures, which are invoked when valve C and D are opened, are added to safety holons *C* and *D*. These procedures fetch the chemical name contained in the tank connected to the valve and inform safety holon *P* of that name.
- Safety holons developed on Windows/JDPS platform can work with application holons developed on a different platform, i.e. Linux/C^{++} platform.

Because JDPS supports reliable broadcast messaging, application objects can send messages regardless of locations and replication degrees of safety holons, and this enables also the dynamic reconfiguration of safety holons, i.e. safety holons can be added, replaced, migrated and replicated without stopping operations. Without JDPS, application holons must stop their operations, in order to change destination addresses of their messages for invoking safety holons.

JDPS objects are implemented as Java threads, and most safety constraint checking mechanisms are simple procedures that do not include any inference process: there-

fore both sizes and computational load of safety holons are not major obstacles for adopting the proposed mechanism. Namely, safety holons have comparable sizes and computational load as usual programs. However, the delay of communication among application holons and safety holons is the disadvantage that cannot be neglected, i.e. in the current implementation, 5 m seconds are required for a single message sending and answer receiving cycle between an application and a safety holons when they are located at the same computer, and usual safety ensuring processes consist of several cycles. Therefore it is applicable to only a system, of which response time is more than 100m seconds. When message exchanging holons are located at different computers, a single message exchanging cycle requires 15m seconds, but in this case, multiple message exchanging cycles can be carried out in parallel, and the proposed mechanism works well without decreasing the performance seriously. Neither the network traffic is a serious issue. In usual applications even in the most pessimistic cases, the number of messages exchanged among holons may be less than the twice of that exchanged in the system without the safety mechanism.

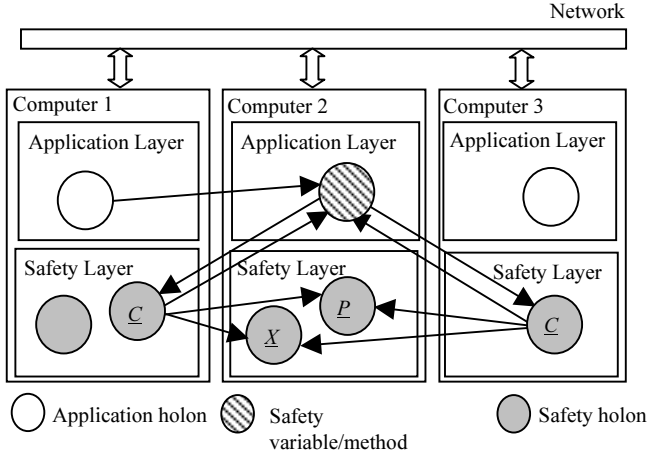


Fig. 5. Implementation of safety holons

4 Conclusion

A mechanism for ensuring safe behaviors of HMSs has been proposed with an experimental system development to make HMSs more practical. The constraints to be imbedded to the proposed mechanism, i.e. safe behavior constraints, constitute the core of the mechanism. These constraints differ depending on applications of course, and therefore the most important issue is the systematization of safety constraints in individual application categories. For example, safety constraints only about tank overflow and chemical mixing were implemented in the experimental system, but

there are many other things to be considered for operating the system safely. This systematization includes also the behavior patterns when safety constraints are not satisfied. The only thing that the experimental system can do, when the safety constraints are violated, is to stop operations, therefore the current implementation is difficult to be applied to systems without operators.

References

1. Leeuwen, E., Norrie, D.: Intelligent manufacturing: holons and holarchies. Manufacturing Engineer, Vol.76, No.2 (1997)
2. Marik, V., et.al. (eds.): Knowledge and Technology Integration in Production and Services. Kluwer Academic Publishers (2002)
3. Deen, S. M. (eds.): Agent-based Manufacturing - Advances in the Holonic Approach -. Springer (2003)
4. Proceedings of the Joint Symposium of HMS and HARMONY Project of IMS Program. The Japan Society for Precision Engineering (2003 in Japanese)
5. Hasegawa, T., Seki, T., Tamura, S.: Environment for developing Holonic Manufacturing System applications - HMS Shell. Proc. of 6th Intl. Conf. on Intelligent Autonomous Systems, Venice, Italy (2000)
6. Imasaki, N., Tharumarajah, A., Tamura, S.: Distributed self-simulation of Holonic Manufacturing Systems. In reference 2 (2002)
7. Tamura, S., Seki, T., Hasegaw, T.: HMS Development and Implementation Environments. In reference 3 (2003)

A Real-Time Interface for Holonic Control Devices

Robert William Brennan[1], Kenwood Hall[2], Vladimír Mařík[3], Francisco Maturana[2], and Douglas Hector Norrie[1]

[1] Department of Mechanical and Manufacturing Engineering
University of Calgary, 2500 University Drive NW
Calgary, Alberta T2N 1N4, Canada
{brennan, norrie}@enme.ucalgary.ca
[2] Rockwell Automation
1 Allen-Bradley Drive
Mayfield Heights, OH, USA, 44124
{Khhall, fpmaturana}@ra.rockwell.com
[3] Rockwell Automation
Americka 22,
Praha 2, Czech Republic 120 00
vmarik@ra.rockwell.com

Abstract. In this paper we present a general specification for a real-time transformation interface for Holonic Control Devices. This interface is intended to provide the logical linkage between the two worlds of agents and machines, which is needed to realise truly holonic systems.

1 Introduction

Agents manipulate information; machines manipulate the physical world. Without an effective real-time interface between these two worlds, agents and machines will continue to exist and operate largely apart as they do today. This paper describes an architecture for this interface based on a third world: that of control. A detailed specification for this architecture is currently being developed by the Holonic Manufacturing Systems Consortium [10].

In the next section, we introduce this real-time interface and contrast it with non-real-time interfaces in multi-agent systems. In section 3, we consider the real-time interface problem in more detail and introduce our layered architecture. This architecture is then described in more detail in section 4. Finally, we conclude with a summary and some thoughts on our future work in this area.

2 Background

The concept of a real-time interface between the soft world of agents and the physical world of machinery is tied closely to some fundamental ideas in holonic systems and holonic manufacturing systems. By definition, “holons” contain both an information processing part and a physical part [10]. Moreover, “holonic systems” are essentially

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 25–34, 2003.

adaptive agent-machine systems. The real-time control interface problem is therefore a concern for those wishing to apply holonic concepts to manufacturing and other areas. Not surprisingly, the Holonic Manufacturing Systems Consortium [12] has a work group devoted to Holonic Control Devices (HCD). Although approaches developed by this group would be applicable to a larger area than manufacturing, an even wider application focus would be desirable.

It is the need to link, even integrate, between the information world and the physical world that distinguishes HMS from the mainstream multi-agent systems community; the physical world of machinery being its special concern. The Foundation for Intelligent Physical Agents [8], formed in 1996, also recognised the need for agent technology to link to the physical world. FIPA however, did not pursue the notion of *physical agents* closely until recently. FIPA now has work groups involved in the manufacturing production and scheduling areas [22]. As the necessity for solving the real-time control interface problem becomes even more apparent, we anticipate that additional research groups will emerge with activity in this area. One thing is certain: there can be no large-scale deployment of adaptive agent-machine systems until the real-time interface problem is solved.

2.1 Holonic Control Devices

As illustrated in Figure 1, the scope of this work is limited to HCDs associated with *holonic agents* [6,17,18] occupying the lowest layer of a manufacturing holarchy. This includes those features necessary to enable the physical holons to enter into *negotiations* about the performance of *manufacturing tasks* and to mutually *coordinate* the performance of those tasks. Such features include: (i) the ability to locate, join, leave and participate in *cooperation domains* for the performance of manufacturing tasks, (ii) the ability to *reason* about manufacturing tasks and their relationships to distributed control applications, and to acquire and share *knowledge* related to such reasoning, and (iii) the ability to issue appropriate *management commands* to dynamically modify existing applications to perform new tasks or to recover from abnormal operations

2.2 Interface Agents

As noted previously, in order to realise truly holonic systems a real-time interface is required to effectively integrate the worlds of agents and machines. The notion of interfacing domains with very disparate world-views and knowledge representations is not new to agent systems. In fact, a considerable amount of research has been conducted on the development of interface agents (IAs) that are typically used to allow people to interact with agent-based systems. Although the specific design of IAs may differ from system to system to system, they tend to share the same basic characteristics as noted by Laurel [15]: (i) agency, (ii) responsiveness, (iii) competence, and (iv) accessibility.

In her definition of “agency”, Laurel notes that interface agents must be “... capable of understanding our needs and goals in relation to them (either explicitly or implicitly), translating those goals into an appropriate set of actions, and delivering

the results in a form that we can use". Although this definition is expressed in terms of a human/agent system interface agent, the definition is still quite useful for our agent/machine interface. For example, the higher-level agents at the organisation level in Figure 1 will express their needs and goals in a much more abstract form than the physical holons are capable of understanding (e.g., "I need to expedite order A35"). These needs and goals must be translated to a set of actions that is understood at the machine level (e.g., "AGV #3 report your current location"), then delivered in a form that the higher-level agents can use (e.g., "order A35's expected completion time is tomorrow at 15:30").

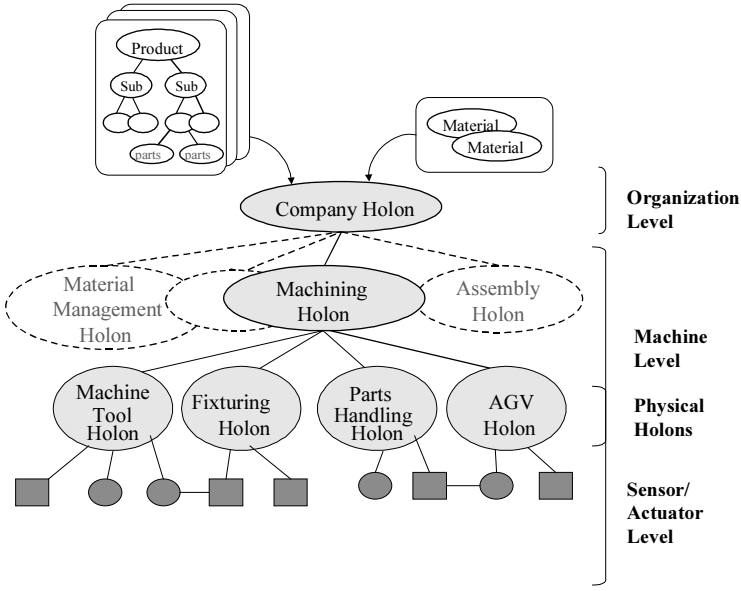


Fig. 1. Position of physical holons in a manufacturing holarchy.

At the machine level, agents have clear temporal restrictions (i.e., both hard and soft real-time constraints), and as a result, must dedicate their time to specific real-time activities. For example, Raja and Lesser [20], classify real-time agent activities into three main activities: domain activities, control activities, and meta-level control activities. Domain activities include the executable primitive actions that achieve the high-level tasks required at the organisation level. Control activities are classified as either those activities that choose the high-level goals and set constraints on how to achieve them (i.e., scheduling activities) or those activities that facilitate cooperation with other agents in order to achieve the high-level goals (i.e., coordination activities). Finally, meta-level control activities are those activities that optimise the agent's performance (e.g., through appropriate processor allocation).

By definition, agents and holons are responsive entities [24,10]. In the case of software agents, responsiveness is typically defined in terms of an agent's ability to respond to the user. However, in holonic systems and HCDs in particular, responsiveness has a deeper meaning; not only are these systems required to respond to the user, they must also respond to events in the physical world of machinery.

Because of the real-time nature of these physical devices, this responsiveness is also defined in terms of the HCD's ability to manage soft and hard real-time tasks that are periodic or aperiodic in nature. Although a considerable amount of work has been conducted in this area for classic, centralised control systems [1], this area is only recently being investigated for real-time distributed control systems [7,21].

Laurel [15] defines "competence" in terms of the domain or environment in which the IA operates and points out that these agents "... must possess (or be able to generate) both meta knowledge and multiple representations". The domain or environment of competence is clearly the domain holonic systems domain of information/physical agents where competence is broadly defined in terms of autonomy and cooperation [10]. More specifically, the meta knowledge (or knowledge about the problem solving domain) HCD interface agents must possess can be thought of in terms of a "process abstraction", or an understanding of the capabilities of the HCD and how these capabilities can be exploited by higher-level agents. As well, if truly dynamic and intelligent fault monitoring and recovery is to be achieved [3], the physical holons must be aware of their capabilities and limitations in this area (e.g., homogeneous and diverse redundancy strategies and fail-safe modes) and also how to access high-level assistance when they are not capable of recovering from a failure.

The interface agent's ability to deal with multiple representations is a key aspect of responsiveness. As Brennan [4] notes, an IA should "... increase the odds that the user and the system will be able to communicate effectively and that ambiguities in one representation will be disambiguated by another ...". Clearly, this is a very important characteristic of our HCD interface, since it is responsible for reconciling the representations of the higher-level agent world and the lower-level control world.

Closely related to this is the requirement for accessibility. Conceptually, this means that the user should be able to predict the IAs behaviour in given situations. On the surface, this is made difficult in an HCD because of the different representations at the agent/control levels. For example, to obtain scheduling information at the organisation level, a group of HCDs may have to be accessed. Each of these holons would only have a partial picture of the schedule and would also be focused on detailed tasks such as feedback control and vision processing. Although the higher-level agents need to predict the final delivery of the product they are tracking, it is not important that they predict the detailed behaviour of each HCD (e.g., the precise trajectory of an AGV).

In the next section we describe a general model for a transformation interface for Holonic Control Devices. The interface agents within this model are intended to address the basic characteristics of interface agents described above and provide these capabilities in a real-time environment.

3 The Transformation Interface

Although there has been a considerable amount of work on holonic and agent-based approaches to the upper (planning and scheduling) level of control [19] very little work has been done on applying these techniques to the lower, real-time control level. In order to integrate these two areas together, a generic interface between the high-

level agents and the low-level control functionality is required. As a result, two basic approaches may be taken to implement the control solutions.

First, higher-level software agents could be integrated with controllers that support extant control functionality such as controllers based on IEC 61131-3 [14]. The role of interface agents in this case would be primarily to provide accessibility. In other words, the interface should allow higher-level software agents to monitor the status of the control functionality and also facilitate these agents in their reasoning about process improvement, and in the case of latent faults, recovery strategies. Similarly, interface agents should allow program units at the control level (e.g., IEC 61131-3 function blocks) to access data in an appropriate format (e.g., set points, deadlines, etc.).

A second approach involves integrating the higher-level software agents with controllers that support distributed real-time control. For example, given the distributed, event-based characteristics of the IEC 61499 model [16], there is the potential to go beyond accessibility to the realisation of a multi-layered, real-time intelligent control system. In particular, the notion of agency at the control level becomes more relevant with the use of IEC 61499 function blocks [2]. For example with this model, some of the fault recovery functionality that would be handled at the higher software agent level could potentially be handled by IEC 61499 holonic agents.

As is shown in Figure 2, both approaches require a transformation interface between the upper-level *software agents* and the lower-level *control functions*. The authors are currently investigating both approaches in order to allow for a generic interface definition. This will permit both current (i.e., IEC 61131-3) and future (i.e., IEC 61499) control specifications to interoperate inside an HCD. This transformation interface will also need to include interactions with simulated physical systems and with provision for adequate security at both the *software agents* and the *control functions* levels.

As a consequence of this initial activity there is a need to establish a migration path from existing systems based on IEC 61131-3 to distributed intelligent control at the physical device level based on IEC 61499. In particular, an IEC 61499-based HCD has the potential to offer well-defined, function block-based open interfaces to the agent-based high-level control layer to allow reconfiguration of the control applications. These applications are portable (runtime code independent) and interoperable through well-defined communication function blocks, not only within one controller but also among diverse controllers. The control applications are linked to the processes via IEC 61499 service interface function blocks (SIFB), which ensures independence from low-level implementation details of communication links and protocols. For more information on the IEC 61499 specification (and function block types), please refer to Lewis [16] or IEC TC65/WG6 [13].

Agent systems are now being standardised through organisations such as FIPA. As well as a generic specification for a transformation interface, there will be the need for a specific specification relative to standards such as FIPA that would allow the software agents to interact in the high-level information domain. In summary, the transformation interface will be composed of three fundamental components: (i) the low-level interface, (ii) simulation interface, and (iii) an interface to agent standards such as FIPA. In the next section, we take a closer look at the HCD transformation interface.

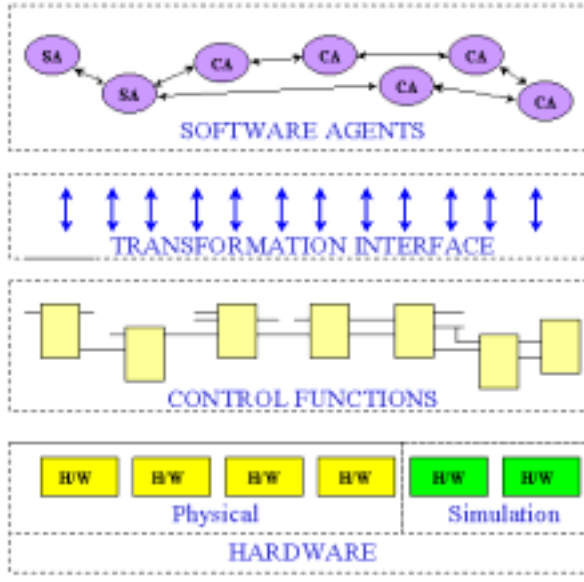


Fig. 2. The HCD transformation interface.

4 The HCD Architecture

Figure 3 shows a more detailed version of the Holonic Control Device Architecture. In this section we describe the different layers and their purpose.

The Deliberative layer is two fold: (i) application domain specific functionality and (ii) generic functionality. The former functionality corresponds to a set of user-defined functions that are made available to the agent/holon instances inside the HCD throughout function calls. The latter functionality corresponds to a set of generic decision-making components that act as the nervous system of the HCD instance for decision making. As is illustrated in Figure 3, there are four generic decision making modules: Planner, Process Model, Execution Control, and Diagnostics.

The *Planner* component carries out the construction of control-action steps for the HCD instance. It also negotiates with other HCD instances, which can be local or remote, via an agent communication language (e.g., FIPA, KQML, etc.). This decision-making module can be thought of as the core component that provides the HCD with "holonic" capability (i.e., autonomous/cooperative behaviour). As well, it is one of the components of the HCD interface architecture that contributes to the HCD meeting Laurel's [15] definition of "agency": i.e., the planner "translat(es) … goals into an appropriate set of actions".

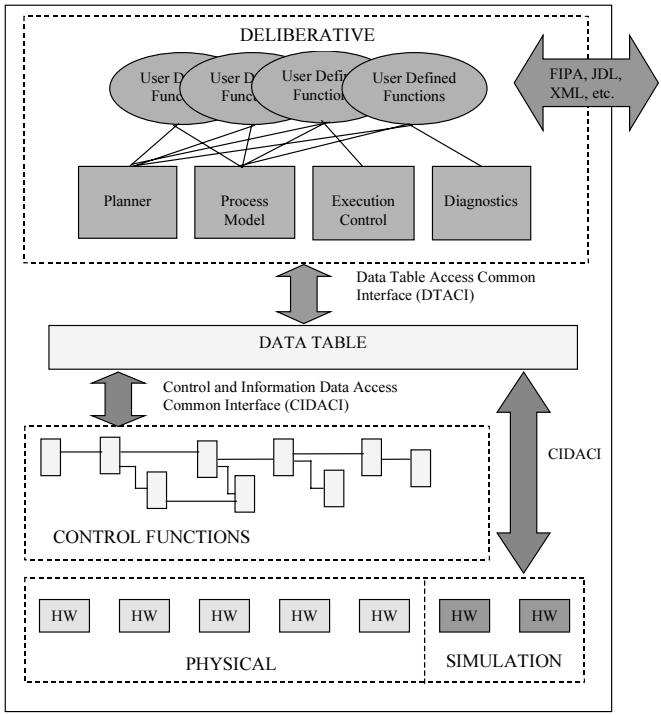


Fig. 3. The HCD architecture.

As was noted previously, an important characteristic of interface agents related to "competence", is that they should possess meta knowledge (i.e., knowledge about the problem solving domain). This is the primary role of the *Process Model*, which carries out a local model simulation (which can be a mathematical expression that quantifies performance metrics) of input data to quantify the current control step performance, thereby evaluating present execution states or forecasting near future states.

The *Execution Control* component coordinates the execution of control action steps thereby carrying out the HCD plans, which are built and committed by the planner during the inter-HCD instance negotiation. This is a further refinement of the planner's functionality, which results in the higher-level goals being translated into a form that can be used at the control level. This interface is not just one-way however (i.e., high-level to low-level); it also reacts to changes of state in the control layer via event monitoring.

Finally, the *Diagnostics* component carries out a local self-assessment activity to verify the correct operation of the hardware attached to the HCD and/or HCD instances. As noted previously, this capability of an HCD interface is closely tied to the deeper meaning of responsiveness of real-time systems. In these systems, correctness is synonymous with timeliness. As a result, it is necessary that we have an element of the HCD that is responsible for verifying that deadlines are met, and if not, that corrective action is taken.

The *Deliberative Layer* communicates with the other layers throughout the device's data table via a Data Table Access Common Interface (DTACI). The *Data Table Layer* is the state data repository for the HCD instances. In its most basic form, the DTACI can be thought of as being similar to input and output image tables of a conventional programmable logic controller (PLC) [23]. Similar to a conventional image table, each HCD instance has a private section of the DTACI memory for storage. However, rather than just providing access to the PLC control logic during the scan cycle, both Deliberative and Control layers can read/write data from/to the DTACI.

The *Control Functions Layer* is the user defined application logic that controls the activities of the physical hardware or process that is attached to the HCD. Although this figure shows IEC 61131-3 function blocks, any of the IEC 61131-3 languages for programmable logic controllers (i.e., IL, ST, LD, SFC, LD) could be used to specify the control logic. As well, distributed intelligent devices at the physical layer can also be supported at this layer by the IEC 61499 model. The control application should be capable of sending agent messages (e.g., ACL) to the deliberative layer.

Finally, the *Physical Layer* represents the hardware (actuators and sensors) and process entities that are controlled by the HCD. The *Simulation Layer* is the simulation of the physical hardware and process entities. This additional layer will allow larger-scale HCD implementations to be tested, and also provide us with the opportunity to interface with recent simulation efforts for benchmarking alternative multi-agent approaches (e.g., the work by Cavalieri [5]).

Both the Physical and Simulation layer communicate with the HCD through a *Control and Information Data Access Common Interface* (CIDACI). Currently, two closely related approaches are being investigated to implement this interface, (i) data table, and (ii) function block adapters. The first approach, shown in Figure 3, involves a shared data table that can be read from and written to by both the deliberative and the control functions layers. This approach was presented at the Holonic Manufacturing Systems HMS-21 plenary meeting [12]. A second approach that is currently being investigated is to use function block adapters (FBA) as proposed by Heverhagen and Tracht [11]. Currently, their approach is intended to provide an interface between Real-time Unified Modeling Language (RT-UML) capsules and IEC 61131-3 function blocks. In other words, RT-UML capsules provide the interface on the object or agent side (i.e., the deliberative layer) while IEC 61131-3 function blocks would provide the interface on the control function side. The advantage of this approach is that, function block adapters in combination with Heverhagen and Tracht's FBA language, provide a means to unambiguously express the interface mapping. Additionally, the FBA approach appears to be particularly well-suited to the IEC 61499 model. This is a result of the close correspondence between the RT-UML capsule stereotype and the IEC 61499 basic function block [9] as well as a potentially simpler implementation with IEC 61499 service interface function blocks on the control side.

5 Summary and Future Work

In this paper, we provided a general overview of a real-time interface architecture for Holonic Control Devices. This real-time interface is a very hard problem and its solution will require both new approaches and adaptation of existing techniques. The hardest part of the problem occurs when configurations of physical machines, components, and their linkages change. Such "dynamic reconfiguration" situations can only be handled with today's techniques when the changes are few, simple, and occur slowly. Truly dynamic reconfiguration in which both agent and machine worlds change rapidly and extensively, with changes in each spurring and influencing consequent changes in the other, are well beyond our existing capability. This problem will need to be solved before we can design and maintain adaptive agent-machine systems on any significant scale.

References

1. Bennett, S. (1994) Real-Time Computer Control: An Introduction, Prentice Hall.
2. Brennan, R.W. and Norrie, D.H. (2001) "Agents, holons and function blocks: distributed intelligent control in manufacturing," Journal of Applied Systems Studies, **2**(1), 1–19.
3. Brennan, R.W. and Norrie, D.H. (2002) "Managing fault monitoring and recovery in distributed real-time control systems," 5th IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, Cancun, Mexico, pp. 247–254.
4. Brennan, S. (1984) "Interface agents", Technical Report, Atari Systems Research Laboratory, Sunnyvale, California.
5. Cavalieri, S., Garetti, M., Macchi, M., and Taisch, M. (2000) "An experimental benchmarking of two multi-agent architectures for production scheduling and control", Computers in Industry, **43**(2), 139–152.
6. Christensen, J.H. (1994) "Holonic manufacturing systems: initial architecture and standards direction, First European Conference on Holonic Manufacturing Systems, Hanover, Germany, pp. 1–20.
7. Douglass, B.P. (1999) Doing Hard Time, Addison-Wesley.
8. FIPA (2003) Foundation for Intelligent Physical Agents Web Site, http://www.fipa.org/.
9. Fletcher, M., Brennan, R.W. and Norrie, D.H. (2001) "Design and evaluation of real-time distributed manufacturing control systems using UML Capsules", 7th International Conference on Object-oriented Information Systems, Springer-Verlag, pp. 382–386.
10. Gruver, W.A., Kotak D.B., van Leeuwen, E.H., Norrie D. (2003) "Holonic manufacturing systems - phase 2", In: Holonic and Multi-Agent Systems for Manufacturing, Springer LNAI 2744, Springer, Heidelberg.
11. Heverhagen, T. and Tracht, R. (2002) "Implementing function block adapters", Lecture Notes in Infomatics, Verlag, pp. 122–134.
12. HMS (2003) Holonic Manufacturing Systems Consortium Web Site, http://hms.ifw.uni-hannover.de/.
13. IEC TC65/WG6 (2000) Voting Draft – Publicly Available Specification - Function Blocks for Industrial Process-measurement and Control Systems, Part 1-Architecture, International Electrotechnical Commission.
14. John, K. and Tiegelkamp, M. (2001) IEC 61131-3: Programming Industrial Automation Systems, Springer.

15. Laurel, B. (1997) “Interface agents: metaphors with character”, In: J.M. Bradshaw (Ed.), Software Agents, The MIT Press, Menlo Park, California.
16. Lewis, R.W. (2001) Modelling Control Systems Using IEC 61499, The Institution of Electrical Engineers.
17. Marik, V., Fletcher, M. and Pechoucek, M. (2002) “Holons and agents: recent developments and mutual impacts”, In: V. Marik, O. Stepankova, H. Krautwurmova, M. Luck (Eds.): Multi-agent Systems and Applications II, Lecture Notes in Artificial Intelligence, No. 2322, Springer-Verlag, Berlin, pp. 233–267.
18. Maturana, F.P., Tichý, P., Šlechta, P., Staron, R.J., Discenzo, F.M., and Hall, K. (2003) “A highly distributed intelligent shipboard system for US Navy mission automation,” In Proceedings of the Thirteenth International Ship Control System Symposium, April 7–9, Orlando, Florida, USA.
19. McFarlane, D.C. and Bussmann, S. (2000) “Developments in holonic production planning and control”, Production Planning and Control, **11**(6), 522–536.
20. Raja, A. and Lesser, V. (2001) “Real-time meta-level control in multi agent systems”, In Adaptability and Embodiment Using Multi-agent Systems Workshop, Czech Technical University.
21. Soler, J., Julian, V., Carrascosa, C., and Botti, V. (2000) “Applying the ARTIS agent architecture to mobile robot control”, In Proceedings of IBERAMIA-2000, Atibaia, Sao Paulo, Brazil, pp. 359–368.
22. Ulieru, M. (2001) “FIPA technology overview: holonic enterprise”, FIPA Inform!, **2**(2), 3–4.
23. Webb, J.W. and Reis, R.A. (1999) Programmable Logic Controllers: Principles and Applications, Prentice Hall.
24. Weiss, G. (1999) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press.

Integration of Automation Resources in Holonic Manufacturing Applications

Paulo Leitão[1], Raymond Boissier[2], Francisco Casais[1], and Francisco Restivo[3]

[1] Polytechnic Institute of Bragança, Quinta Santa Apolónia, Apartado 134,
P-5301-857 Bragança, Portugal,
{pleitao, fcasais}@ipb.pt
[2] Université Paris-Nord, GRPI.
93206 Saint Denis Cedex 1, France,
boissier@iut-stdenis.univ-paris13.fr
[3] Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias,
P-4200-465 Porto, Portugal,
fjr@fe.up.pt

Abstract. Holonic and agent-based paradigms are very suitable in the development of distributed manufacturing control systems, taking advantage of their modularity, decentralization, and ability to support dynamic and complex system design features. However, the integration of manufacturing resources within the holonic manufacturing control applications remains a problem, because no efficient standard allows an easy, transparent and essentially independent integration. This paper proposes an integration process that allows the integration of automation resources independently from the holonic control application, using some concepts derived from the Manufacturing Message Specification (MMS) application protocol and implemented over a distributed object platform.

1 Introduction

Nowadays, in order to face stochastic and volatile environments, manufacturing systems must exhibit increasing agility. This implies the corresponding control application must also adapt to the occurrence of unexpected disturbances, very likely through dynamic and distributed structures.

The Holonic Manufacturing System (HMS) paradigm seems a promising approach to support these actual and emergent requirements. HMS translates to the manufacturing world systemic concepts developed by A. Koestler concerning living organisms and social organizations, mainly that complex systems are hierarchical systems formed by intermediate stable forms, being simultaneously a part and a whole [1]. In industry, the word holon, which illustrates this hybrid nature, represents the manufacturing components and activities, such as machines, products and parts. Their behavior is determined by their cooperation with other holons, as opposed to being determined by a centralized mechanism.

Due to the heterogeneous manufacturing environment, it is hard and cumbersome to develop holonic manufacturing applications that integrate manufacturing resources, because the holonic application is highly dependent of the

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 35–46, 2003.

resource interfaces, normally developed one-of-a-kind. The solution requires a standardized integration process that makes transparent the interface between the holonic manufacturing applications and the manufacturing resources. Some relevant approaches, such as the MMS protocol and the OPC (OLE for Process Control), do not cover integrally the manufacturing resource characteristics and complexity, and do not support the independency between the control and the integration domains.

Our approach to the problem is the re-use of the basic MMS concepts over a distributed object based platform, thus forming a set of objects that are invoked remotely by the holonic control application. These objects are always the same on the client side (i.e. the controlling application), but are customized in the server side (i.e. the controlled programmable manufacturing device), according with the resource details.

In this paper, initially a holonic manufacturing control architecture is briefly described, which demands for a transparent resource integration in order to support heterogeneity and interoperability. In section 3 an overview of available technologies to support the resource integration is presented, while section 4 describes the proposed resource integration mechanism based in the virtual resource concept and in a client/server distributed object platform. In the last part, the implementation of two virtual resources, one for a programmable logic controller (PLC) and other for an industrial robot, that validates the proposed approach, is described.

2 Holonic Manufacturing Control System

The ADACOR (Adaptive Holonic Control Architecture for Distributed Manufacturing Systems) architecture, proposes a new holonic approach for flexible manufacturing systems control, focused on distributed manufacturing shop floor control, and facing the dynamic and agile adaptation to disturbances. The architecture is based on a set of autonomous, intelligent and co-operative entities, designated by holons, to represent the automation factory components, aiming to support the distribution of skills and knowledge. These manufacturing components can be both physical resources (numerical control machines, robots, programmable controllers, etc) and logic entities (products, orders, etc), grouped in the following main types of holons: *product*, *task*, *operational* and *supervisor* [2]. The operational holon type represents the physical resources and comprise the physical manufacturing resource, capable of performing manufacturing operations and the Logical Control Device (LCD), which acts as an agent, and contains*: inter-holon interaction mechanisms* to support negotiation and coordination with other holons, *manufacturing control functions* that regulate the behavior of the holon aiming to pursuit its goals, and *intra-holon interaction mechanisms* to support the interaction between the physical manufacturing resources and the LCD.

The internal architecture for a generic LCD belonging to an ADACOR holon, Fig. 1, comprises a local knowledge base and three main components: *decision* (DeC), *communication* (ComC) and *physical interface* (PIC) components [3].

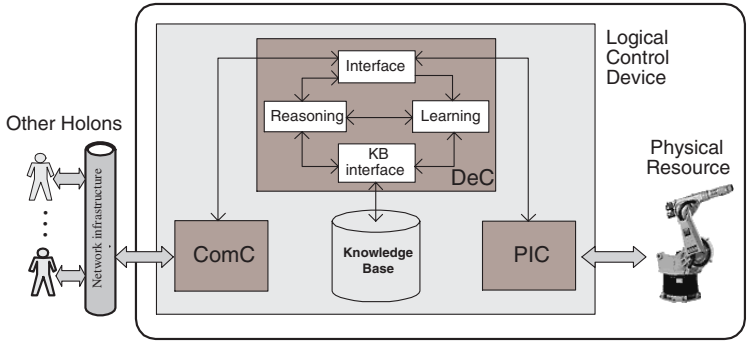


Fig. 1. Internal Architecture for an ADACOR Agent

The *physical interface component* provides the mechanisms to integrate the manufacturing resources. Since the manufacturing factory is a heterogeneous environment, with the distributed holons and the automation devices placed on a wide variety of interoperable control platforms, a crucial point when holonifying manufacturing components, such as robots and machine-tools, is the connection between the software part of the holon and the physical manufacturing resource. As the local resource controllers have mostly closed architectures it is necessary to develop wrappers to hide the details of each resource controller and supplies primitives that represent the functionality of the physical manufacturing resource [4]. Thus, the PIC component comprises the mechanisms for the interaction with the physical devices that makes transparent the access to manufacturing resources from the holonic control application, and independent the control application from the integration domain.

3 Manufacturing Resource Integration Technologies

As referred, the integration of manufacturing resources into holonic and agent-based applications assumes a crucial aspect, requiring mechanisms that make transparent and independent the control application from the details of local resource controllers. Moreover, FIPA (Foundation for Intelligent Physical Agents) [5], which aims to produce standards for the interoperation of heterogeneous software agents, does not present, at the moment, specifications to support the integration of physical resources, in spite of the effort to introduce new specifications that support manufacturing requirements, through the FIPA Product Design and Manufacturing working group.

MMS, which defines the application layer of the ancient MAP (Manufacturing Automation Protocol) protocol, provides a platform capable to interconnect various industrial devices supplied by different suppliers. MMS brought together many IT and manufacturing specialists to define a common framework for developing communication support between industrial computerized equipment, under the ISO 9506 international standard [6]. The basic concepts of the MMS protocol are a client-server mechanism and the VMD (Virtual Manufacturing Device) model. The VMD, associated with every real manufacturing device, is an abstract model of the server application, which maps the functionalities of the real device, and offers all services concerning itself and its related abstractions, mainly: *domains*, *variables*, *program invocations* and *events*. This set of objects and generic services can be applied to a large set of manufacturing devices, such as robots and numerical control machines [7]. However, the technology vendors do not closely follow the MMS standard, since certain functionalities have different implementations depending of the machine vendor and the underlying network. This missed adhesion by the vendors to the standardization associated to the high price of this technology retracted the expansion of the MMS technology in the market.

Some research teams introduced the idea to use the MMS concepts combined with a distributed object platform technology to integrate the manufacturing resources. The approaches presented in [8,9] use the MMS concept over the CORBA (Common Object Request Broker Architecture) distributed object technology, with successful results. The real-time constraints in the industrial manufacturing world requires real-time response, being the CORBA technology and the classical TCP/IP not adequate. The ReTINA model has been used within the *Jonathan* distributed environment [10], in order to support applications subject to real-time functioning [11].

Another available technology is the OPC, which is based on Microsoft's OLE/COM technology. It allows software in the form of software components to interoperate regardless of where they are located [4]. The OPC servers, OLE/DCOM compliant, offer an automation interface, which allows to design PC-based clients that import real time automation data using standard Windows applications. The Windows proprietary scope remains an important limitation of this approach for the heterogeneous environments; however, some available tools, such as J-Integra [12] and Bridge2Java [13], allow to overcome this problem.

IEC 61499 standard [14,15] is an approach for the easy and quickly integration of large re-configurable systems, defining a way to model the control and execution of algorithms in distributed control systems, being encapsulated, reusable software modules. Within this model, the ancient function block concept is re-introduced in order to make a clear distinction between the event and the triggered algorithms, making easier the verification of time properties [16]. A function block, which is the fundamental unit of software encapsulation and reuse in IEC 61499, encapsulates the control algorithm with physical interfaces, communications, human interfaces, monitoring and diagnostics, and information technology-based services.

At the moment some low level programmable controllers, such as PIC's and PLC's already support the IEC 61499 standard, which makes adequate this approach for the resource integration by the direct communication between the entities. However, for the low level programmable controllers that do not support yet IEC61499, and essentially for the communication between high level programmable controllers such as robots, numerical control machines and PLC's, which is the main focus of the paper, IEC 61499 is not yet a solution for the resource integration.

From the preceding, it is clear there is a need for a low cost approach that could support transparent interfaces for physical manufacturing resources, and allows easy integration of these resources into holonic control applications. The use of light MMS concepts combined with distributed object paradigms seems a suitable approach to make transparent the resource integration from the agent-based or holonic control system.

4 Resource Integration Approach

Our ADACOR approach to transparent resource integration within holons is taking advantage of the OO-MMS mechanism for communication between any client and a Virtual Machine Device server [8]. The scheme is displayed in Fig.2.

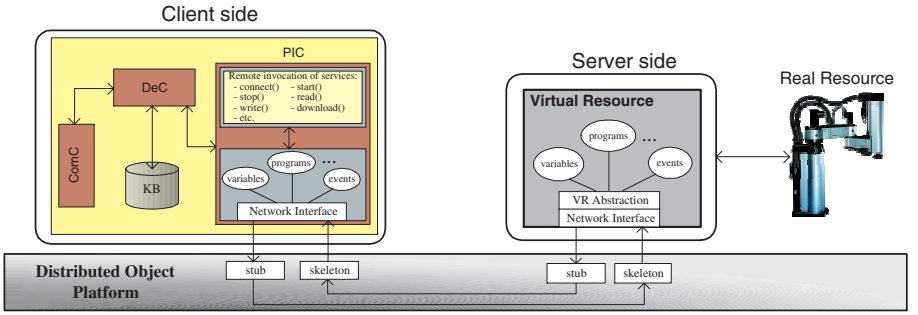


Fig. 2. Invocation of Remote Services using the Virtual Resource concept

Next, the main concepts of the schema, mainly the virtual resource and the client-server model, will be deeply analyzed.

4.1 Virtual Resource

The server part in the proposed mechanism is the virtual resource, inspired by the VMD concept from the MMS protocol [6]. It acts as an abstract machine that represent the functionality of the real manufacturing device and its local

controller, supplying primitives to be invoked remotely by the client part. The virtual resource components can be re-used for additional and new applications, since the manufacturing resources are independent from the control application.

The virtual resource is developed for each physical device according the specifications of the machine vendors and comprises a set of objects that maps the services of manufacturing resources. The use of MMS specifications in the definition of services is important in order to standardize the approach, but due to the complexity of these specifications, a sub-group of services were defined, closest as possible to the MMS specifications, in order to make things easier and lighter. These services are grouped in re-use libraries, such as the *VR Support*, *Variable Handling*, *Program Handling* and *Events*, as represented in Fig. 3, which shows some services that provide the interaction with the physical manufacturing devices.

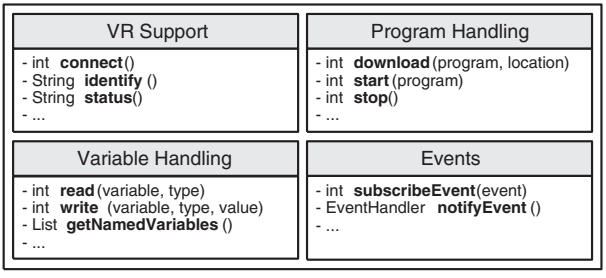


Fig. 3. Re-use Libraries of Services Provided by the Virtual Resource

The objective, input parameters and return values of available services are always the same, making transparent the development of the holonic or agent-based manufacturing applications from the particular details of each resource, improving the ability to support the heterogeneity.

The interaction between the physical manufacturing resource and the virtual resource is also dependent of the communication platform, such as serial link, fieldbus networks and TCP/IP protocol or different connectivity's applications developed under OPC technology, ActiveX components, etc.

4.2 Client-Server Model

The second main concept in the proposed mechanism is the client-server model. The LCD device acts as a client part accessing to the real manufacturing resource by invoking remotely the primitives that represent services in physical resource.

The industrial manufacturing environments are characterized by its heterogeneity, with the distributed processing resources, i.e. computers, industrial controllers and automation devices, running in distinct platforms, such as Windows, Linux and AS400. This heterogeneity requires the use of distributed object platforms to support the interoperability between the clients (operational agents)

and virtual resource components. The available technologies to support the distributed object platform are mainly the CORBA, DCOM (Distributed Common Object Model) and RMI (Remote Method Invocation) [4].

CORBA is based essentially in the Object Request Broker (ORB) concept (also designated as software bus or middleware), which allows a local client to invoke methods on a remote platform as if it were local. In order to mask remoteness and networking details, the middleware platform installs end points (stub and skeleton) on the client side and on the server side. The behavior is very close to the Remote Procedure Calls (RPC) mechanism; however, a RPC is offered by a dedicated server, while ORB methods are attached to a client and a server can handle many client. This mechanism requires an independent Interface Definition Language (IDL) for describing interfaces and generating stubs for various target languages, and an object registering mechanism and object locating schemes for unambiguous referencing and easy object access. Java IDL, which is part of Java 2 platform, allows IDL specifications to be compiled into Java interfaces so that java programs can work with a CORBA compliant ORB. Java IDL enables distributed Java applications to transparently invoke operations on remote network services using the industry standard OMG IDL (Object Management Group Interface Definition Language) and IIOP (Internet Inter-ORB Protocol) defined by the OMG consortium. The main advantage of CORBA is to allow object interaction independently of the source language and the execution platform.

Like CORBA, the Java Remote Method Invocation (RMI) is conceptually similar to the RPC, providing a means of communicating between Java applications using normal method calls, and offering the capability for applications to run on different computers. The RMI uses also skeletons to connect the server to the RMI framework, stubs that act as a proxy server in the client's environment, and a registring mechanism to store the location and name of the server object. The major advantages of RMI are its better performance and instead of using an *idl* file as the interface, it uses a normal java class as interface allowing to pass any java object as arguments.

IBM and Sun, with the cooperation of the OMG, jointly developed RMI over IIOP, so called RMI-IIOP, which joined together the interoperability of CORBA and the easy development of RMI. The implementation of the interface platform using RMI-IIOP was the easiest, being necessary to execute two main actions: compile the RMI code with the *-iiop* option, which generates the *stub* and *tie* components, and to start the naming service, using the same procedure as for the CORBA implementation.

The choice of the distributed object platform should take in consideration the easy mapping of MMS-based services, the easy integration with programming environment, the ability to support heterogeneity and the real-time constraints. In the experimental implementation section several platforms will be tested and a comparative analysis will be made.

5 Experimental Implementation

Our approach is validated through the integration of two different automation resources within a generic operational holon: a CPM1 PLC from Omron, which is accessed by a RS232 asynchronous line, and an industrial robot IRB1400 from ABB, accessed through TCP/IP using an ActiveX component.

In order to make easier the access to physical manufacturing resources, our goal is to have a common client, which is the operational holon, independent from the resource controller details and using the same generic methods to access to the automation resource services, such as the start, stop and read methods.

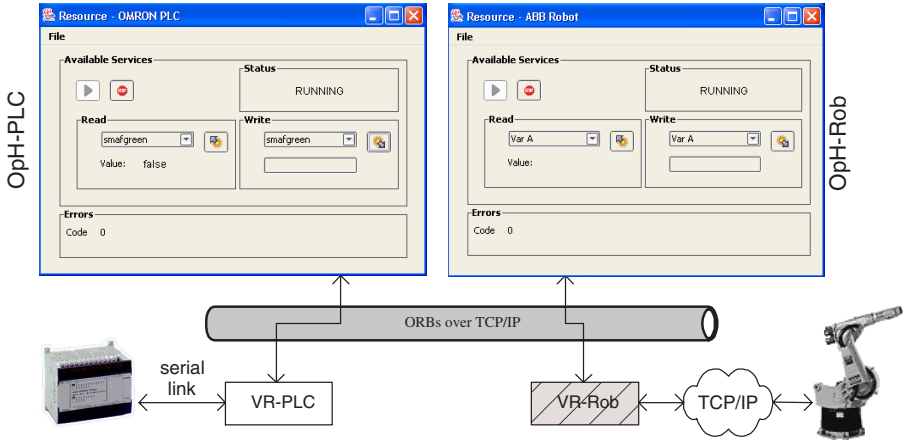


Fig. 4. Conceptual Architecture for the Prototype

The prototype, as illustrated in Fig. 4, uses a heterogeneous system environment, and comprises two virtual resources running in Windows XP platforms and customized for each physical resource, one client running in the Windows 2000 platform and another client running in a Linux platform. In order to test the interface between the client and virtual resource components were tested different approaches, namely using the CORBA, RMI and RMI-IIOP platforms.

5.1 Development of Virtual Resources

The development of virtual resources, a task for integration specialists, encompasses, for each manufacturing resource, the implementation of the methods available on the client side and described in section 4.1. The client ignores the details of this implementation and each developed virtual resource can be re-used by other similar resources or other holonic control applications.

Since it is intended to integrate two different automation resources, two virtual resources were developed, one for the PLC and another one for the industrial

robot. These two servers implement the same services in a different way according the specificities of each resource. In order to illustrate the proposed approach, the implementation of the start service in both virtual resources will be described. In the client side, whatever the resource to be accessed, the invocation always conforms the same template and looks like this:

```
int ret=resource.start(programName);
```

where *resource* is the identifier of the virtual resource that represents the real automation device that is intend to access.

The virtual resource for the programmable logic controller CPM1 is developed according the communication protocol defined by the device [17] and using the serial link for the physical communication with the PLC. The implementation uses the *javax.comm* package, available at `http://java.sun.com`, to support the communication between a java application and an automation device. The code related to the implementation of the start service is illustrated bellow.

```
public int start(String progName){
   int returnCode=-1;
   ...
   returnCode=Write2PLC("@00SC03");
   return(returnCode);
}
```

The primitive essentially writes to the PLC a string containing the *run* command using the *Write2PLC* method, which comprises the message sending to the resource and the wait for the return code. The primitive returns the result of the command execution, returning null in case of success, or a positive number in case of an error.

The development of the virtual resource for the industrial robot was harder than in the previous case, mainly because of the manipulation of the ActiveX component [18]. Since the ActiveX components are adequate for Windows environments, it was necessary to convert the ActiveX component to a java package, using for this purpose the Bridge2Java tool [13]. The start service for the industrial robot is summarised bellow.

```
public int start(String progName){
   ...
   try {
      returnCode=h.s4Run ();
      returnCode=h.s4ProgramLoad(prgID,progName);
      returnCode=h.s4Start(prgID,procedure,nOfCycle,runMode);
   }
   catch(IOException ioe){returnCode=determineErrorType();}
   return (returnCode);
}
```

After the declaration of variables, three commands are executed: *s4Run* that turns on the robot motors, the *s4ProgramLoad* that loads a specified program

to the robot controller and the *s4Start* that starts the execution of the loaded program. The primitive returns null in case of success or a positive integer in case of an error.

5.2 Analysis of Experimental Implementation

Based in the experimental implementation it is possible to extract some conclusions about the proposed approach for the resource integration. First of all, the resource integration problem become easier since the same holon can access to different manufacturing resources without the need of re-design and re-program, increasing the independency between the control and integration domains. The change or modification in a specific manufacturing resource environment does not affect the control application domain, which continues to invoke the services in the same way.

The second advantage is concerned to the easy development of the virtual resource by integrators and factory automation specialists, which only concentrates in the resource controller details and communication platform, without the need to know details about the control domain.

The third advantage is related to the ability to support heterogeneous environments due to the client-server model. During the experimental implementation, different distributed object platforms where tested. The results for the execution of the *read* service are summarized in Table 1. In Table 1 the VR parameter is concerned to the time spent by the virtual resource to execute the specified service, and the C-S parameter is related to the time spent in the interaction between the client and the server. From the experimental results it is

Table 1. Experimental Results

		CORBA(ms)	RMI(ms)	RMI-IIOP(ms)
OpH-PLC	*VR*	41.18	41.03	41.86
	C-S	3.43	2.13	3.31
OpH-Rob	*VR*	12.11	11.75	12.5
	C-S	3.41	2.24	3.30

possible to extract some conclusions related to the behavior of the interface platforms. First, it is possible to verify the independency between the automation devices and the interface platforms: the execution of the service at the virtual resource is independent from the interface platform, and the interaction between the client and the server for each interface platform is independent from the type of automation device. Next, it is possible to compare the different interface platforms. CORBA presents better interoperability between ORB vendors than the RMI interface, since the latter is a Java-to-Java mechanism that limits the

scope of its application. On the other hand, RMI presents the easiest interface development and better communication performance, illustrated by the C-S parameter in Table 1. RMI-IIOP presents an intermediate value of communication performance, but overcome the RMI interoperability problems.

6 Conclusions

The heterogeneity of industrial manufacturing environments is one of the most important challenge for the distributed manufacturing control systems. The integration of manufacturing resources with the holonic manufacturing control applications remains a problem, because no efficient standard allows an easy, transparent and essentially independent integration.

This paper has proposed an integration approach relying on a unified object-oriented model of the various manufacturing resources, in order to give independence to the holonic or agent-based control applications. The virtual resource concept and the client-server model, inspired by the MMS standard, have been used. In this way, the resource integration problem is reduced to the customization of the server side, where it is necessary to develop virtual resources according to each manufacturing resource specifications and details, which will supply primitives to be invoked remotely by the operational holons.

The proposed approach has been validated: connections between a holonic application and two industrial devices with very different specifications and communication protocols (a programmable logic controller and a robot) were realized and the performances evaluated.

In future work, guidelines for systematic development of virtual resources will be developed, particularly in cases with non standardized protocol.

References

1. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L. and Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. In: Computers In Industry, 37, (1998) 255–274.
2. Leitão, P. Restivo, F.: A Holonic Control Approach for Distributed Manufacturing. In: Marik, V., Camarinha-Matos, L., Afsarmanesh, H. (eds.): Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle. Kluwer Academic Press, ISBN 1-4020-7211-1, (2002) 263–270.
3. Leitão, P. and Restivo, F.: Agent-based Holonic Production Control. In: Proceedings of 3^{rd} International Workshop on Industrial Applications of Holonic and Multi-Agent Systems, Aix en Provence, France, 2–6 September (2002) 589–593.
4. Barata, J., Camarinha-Matos, L., Boissier, R., Leitão, P., Restivo, F. and Raddadi, M.: Integrated and Distributed Manufacturing, a Multi-agent Perspective. In: Proceedings of 3^{rd} Workshop on European Scientific and Industrial Collaboration, Enschede, Netherlands, 27–29 June (2001) 145–156.
5. Foundation for Intelligent Physical Agents, http://www.fipa.org/, (May 2003).

6. ISO/IEC 9506-1: Industrial Automation Systems – Manufacturing Message Specification, Part 1 – Service Definition, (1992).
7. Nussbaumer, H.: Téléinformatique IV. Collection Informatique, Presses Polytechniques Romandes (1991).
8. Boissier, R., Gressier-Soudan, E., Laurent, A. and Seinturier, L.: Enhancing numerical controllers, using MMS concepts and a CORBA-based software bus. In: International Journal of Computer Integrated Manufacturing, **14**(6), (2001) 560–569.
9. Ariza, T., Fernández, F. and Rubio, F.: Implementing a Virtual Manufacturing Device for MMS/CORBA. In: Proceedings of 8^{th} IEEE International Conference on Emerging Technologies and Factory Automation, vol. 2, Nice, France 15–18 October (2001) 711-715.
10. Dumant, B., Horn, F., Tran, F. and Stéfani, J.-B.: Jonathan: An Open Distributed Processing Environment in Java.In: Davies, N., Raymond, K. and Seitz, J. (eds.): Midlleware'98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing. Springer, ISBN 1-85-233-088-0, (1998) 175–190.
11. Boissier, R., Epivent, M., Gressier-Soudan, E., Horn, F. Laurent, A. and Razafindramary, D.: Providing Real-Time Object Oriented Industriel Messaging Services. In: Proceedings of European Conference on Object Oriented Programming, Brussels, July 1998.
12. Intrinsyc Software, J-Integra tool. Available on http://www.intrinsyc.com, (May 2003).
13. IBM alphaWorks: emerging technologies. Bridge2Java tool. Available on http://www.alphaworks.ibm.com/tech/bridge2java/, (May 2003).
14. Christensen J.H.: IEC 61499 Architecture : Engineering methodologies and software tools. In: Marik, V., Camarinha-Matos, L., Afsarmanesh, H. (eds.): Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle. Kluwer Academic Press, ISBN 1-4020-7211-1, (2002) 221–228.
15. Balasubramanian S., Brennan R.W. and Norrie D.H.: An architecture for metamorphic control of holonic manufacturing systems. In: Computers in Industry, **46**(1), (2001) 13–31.
16. Vyatkin V. and Hanisch H.M.: Verification of Distributed Control Systems in Intelligent Manufacturing. In: Journal of Intelligent Manufacturing, **14**(1), (2003) 123–136.
17. Sysmac CQM1/CPM1 Programmable Controllers, Programming Manual. Omron, (1996).
18. RobComm User's Guide, version 3.0/3. ABB Flexible Automation, (1999).

JAVA-Based Agent Platform Evaluation

Pavel Vrba

Rockwell Automation Research Center, Americká 22
120 00 Prague, Czech Republic
pvrba@ra.rockwell.com

Abstract. The paper presents a comparison of available JAVA-based agent development tools (platforms) with respect to the specific requirements of agent-based manufacturing control solutions. We discuss the use of agents, as high-level decision-making entities, in combination with the low-level real-time control based on IEC 1131-3 or IEC 61499 standards. The need to run agents, written in JAVA language, as well as the agent platform runtime environment inside existing PLC-based automation controllers is stressed. From this viewpoint, we identify particular attributes which the agent platform should fulfill, like FIPA interoperability, small memory footprint, cost, and security. For selected agent platforms – JADE, FIPA-OS, ZEUS and JACK – we present the results of benchmarking aimed at the speed of the message sending among agents, which might be a crucial property in real-time applications.

1 Introduction

The multi-agent technology has been recognized as a key concept in building a new generation of highly distributed, intelligent, self-organizing and robust manufacturing control solutions [16]. The traditional centralized approaches used so far fail in meeting the steadily increasing requirements for so-called *flexible* or *adaptable manufacturing*. This particularly means the capabilities such as a failure detection of a system part and a safe recovery that minimizes the impact on the functionality of the whole system. Another important attribute is the adaptability to changes in the manufacturing environment required especially in the high-variety low-volume manufacturing. Here the continually changing requirements on the type of the product to be manufactured (personalized products e.g.) along with introducing new production methods over time require quick changes to the manufacturing process configuration. It includes for instance the change to the shop-floor layout, removal of some machines and replacing them with those with new capabilities etc.

Over the past ten years researches attempted to apply the agent technology to various manufacturing areas such as supply chain management, manufacturing planning, scheduling and execution control. This effort has led to the development of a new concept, the *Holonic Manufacturing Systems* (HMS) [16], based on the ideas of *holons* presented by Koestler [11] and strongly influenced by the requirements of industrial control. Holons as basic building blocks of holonic systems are defined as *autonomous* and *cooperative* units with decentralized control [16]. The autonomy

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 47–58, 2003.

means the ability of managing the "local behavior" – each holon is equipped with a knowledge about its local goals which are then solved during the routine operation locally without centralized control. However, in some situations a holon may not be able to accomplish a particular task – in such a case the holon can ask for cooperation the other holons. Inter-holon communication also occurs for example when a global plan of a factory changes and thus the assignment of new local tasks to holons is needed.

At the lowest RT-control level, the main characteristics of holons is their linkage to the physical manufacturing devices – these holons have to be able to read data from sensors and send control signals to actuators. Currently, the low-level RT control is usually carried out by IEC 1131-3 programs (mainly ladder logic code) running on industrial PLC-based automation controllers. In the future, it is planned to replace this standard with a new one developed within the holonic research – the IEC 61499 known as *function blocks* (extension of the IEC 1131-3 function blocks). Since both standards fit well RT control purposes, non of them address the higher level aspects of holons acting as cooperative entities capable of communication, negotiation and high-level decision making. This is the reason why the multi-agent systems technology is currently beeing widely adopted by the holonic community [12], naturally shifting its attention to emerging *agent-based manufacturing systems* [3].

As one of the first results, a general architecture, shown in Figure 1, that encapsulates the low-level RT control subsystem with high-level agent component into a single structure, called *holonic agent*, has been presented in [13].

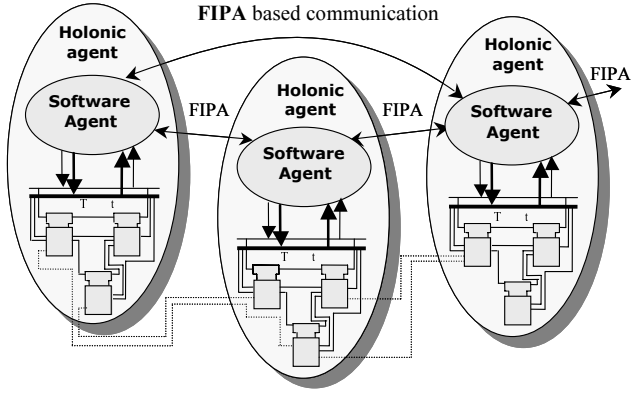


Fig. 1. Holonic agent: combination of software agent and low-level RT control application

However they should be encapsulated into one structure, in majority of current testbed implementations the software agent part of the holonic agent runs apart of the RT control subsystem. Usually, the agent parts are implemented in a high-level programming language (C++ or JAVA) and run on a PC, while the low-level control subsystems implemented in IEC 1131-3 or IEC 61499 run in a PLC-based controllers. As the interface between these two subsystems, a *blackboard* mechanism has been

introduced by McFarlane at al. [14], while others [7] proposed using a special management service interface function blocks.

It is obvious, that for real industrial deployment, particularly where high-degree of robustness is required, the use of PC(s) for running agent-components of holonic agents is not safe and also not possibly feasible for certain types of control systems. The only acceptable solution is to run holonic agents as wholes directly within PLC-based controllers. One controller can host one or more holonic agents, but not all of them – they have to be distributed in reasonable groups over several controllers and allowed to communicate with each other either within a single controller or among different controllers. It is expected, that the inter-holon communication will be standardized. The main reason is that these holonic agents could be involved in global communities of company agents where they can e.g. directly participate in production planning or supply chain management negotiations. The communication standards provided by the FIPA organization [8] seem to be the best suitable candidate for implementing the inter-holonic communication.

The major issue of such a solution is to extend the current architecture of PLC controllers in such a way that they are able to run software agents written in a high-level programming language in parallel with the low-level control code and also provide the interface for interactions between these two layers. The programming language in which the software agents should be implemented can either be C++ or JAVA. However, there are many reasons to prefer the JAVA language to be the target one. One of its advantages is the portability of JAVA programs which the user develops independently of hardware platforms or operating systems – the same application can run either on a PC with Microsoft Windows or Unix/Linux or on a small device like personal digital assistant (PDA) or a mobile phone with Windows CE, Symbian or other operating system with JAVA support. Another reason to choose JAVA is that currently there is a large number of JAVA-based agent development tools available, either as commercial products or also as open-source projects. Moreover some of them are fully compliant with the FIPA standards.

2 Agent Development Tools Characteristics

Basically, the agent development tool, often called an *agent platform*, provides the user with a set of JAVA libraries for specification of user agent classes with specific attributes and behaviors. A kind of a runtime environment that is provided by the agent platform is then used to actually run the agent application. This runtime environment, implemented in JAVA as well, particularly ensures transport of messages among agents, registration and deregistration of agents in the community (white pages services) and also registration and lookup for services provided by agents themselves (yellow pages services). Some other optional tools can also be part of the agent platform runtime, for instance a graphical viewer of messages sent among agents etc.

The implementation of JAVA-based agents into the automation controllers obviously requires such an agent platform runtime being embedded into the controller architecture. Since used as a background for the *real-time* holonic agents, there are specific requirements on the properties of the agent platform, such as a speed,

memory footprint, reliability etc. The evaluation of available JAVA agent platforms, presented in this paper, has been conducted in order to find out to what extent they fulfill these criteria and therefore which ones are the best suitable for the purposes of manufacturing control.

2.1 FIPA Compliancy

Compliancy with the FIPA standards has been recognized as a crucial property ensuring the interoperability of holonic agents not only at the lowest real-time control level (allowing e.g. communication of different kinds of holonic agents hosted by PLC controllers from different vendors) but also the interoperability between holonic agents and other agents at higher levels of information processing within the company, e.g. data-mining agents, ERP agents, supply chain management agents and so on.

FIPA specifications address several aspects of agent systems development, not only the *inter-agent communication* that follows the Agent Communication Language [4] based on the speech act theory [15]. How the agents should be organized and managed within the agent community is covered by the *agent management* specifications. Two mandatory agents have to exist in each community for this purpose. The AMS (Agent Management System) agent provides the white pages services, i.e. holds the names and contact addresses of all existing agents. The DF (Directory Facilitator) agent serves as yellow pages services, i.e. the agents can register their own services with DF that they offer to the community and then the other agents can look up for those with particular services. The FIPA specification of the *message transport protocol* (MTP) defines how the messages should be delivered among agents within the same agent community and particularly between different communities. For the latter case, the protocol based on IIOP or HTTP ensures the full interoperability between different agent platform implementations. It means that the agent running e.g. on the JADE agent platform can easily communicate with agent hosted by the FIPA-OS platform etc.

2.2 Costs

From the cost point of view, the agent platforms that are currently available can basically be divided into two categories – free and commercial ones. Majority of the free agent platforms are distributed under a kind of an *open source* license (e.g. GNU Lesser General Public License) which means that you are provided with the source codes and allowed to modify them. This is an important characteristics since the integration of the agent platform into the PLC-based controllers certainly requires some modifications to be done, e.g. due to different version of JAVA virtual machine supported by the controller, the specifics of the TCP/IP communication support or other possible issues and limitations.

On the other hand, in the case of commercial products the cost in order of thousands USD per each installation e.g. can considerably increase a total cost of the agent-based control solution where a large number of PLC controllers, PCs and possibly other devices running agents are expected to be deployed. Moreover, the

source codes are not available so that all modifications of the platform that needed to be done in order to port it to another device has to be committed to the company developing the agent platform.

2.3 Memory Requirements

An issue that have to be taken into account is usually a limited memory available for user applications on the controller. Within the RAM memory of the controller, which can e.g. be about 4 to 8 MB, the agent platform runtime environment, the agents themselves and also the low-level control code (ladder logic) have to fit inside. There are also smaller PLC-like devices that can have only 256KB of memory available, what would be a strong limitation factor for integrating the runtime part of the agent platform. Fortunately, the agent platform developers, especially in the telecommunication area, are seriously interested in deploying agents on small devices like mobile phones or personal digital assistants (PDAs), that is on devices with similar memory limitations. Due to this fact, for some of the agent platforms their *lightweight* versions have been developed, usually implemented in Java2 Micro Edition (CLDC/MIDP) [9]. It has been documented [2], that the memory footprint of such an agent platform runtime can be less than 100KB, i.e. small enough to fit well within the memory capacity limits of majority of small mobile devices and thus the PLC-based automation controllers as well.

2.4 Message Sending Speed

The last factor considered in this evaluation is the speed of the message sending between the agents. It has already been argued that the holonic agents are expected to be used for real-time control applications where a fast reaction can be a vital characteristics. In Figure 1, a direct communication channel between RT control subsystems of neighboring holons is conceded but it obviously breaks the autonomy of holonic agents. If we are not willing to accept such a violation, communication at the agent level is the only allowable way of interaction among holonic agents. Thus the agent platform runtime, carrying out such interactions, should be fast enough to ensure reasonable message delivery times (i.e. in the order of milliseconds or tens of milliseconds).

We have conducted a series of tests to compare the message sending speed of different agent platforms. Detailed information about the benchmarking testbed configuration and the speed measuring results can be found in Section 3.

2.5 Summary

Table 1 gives an overview of majority of currently available agent development tools with respect to the properties discussed in previous paragraphs. A *security* attribute has been added as a property of the agent platform ensuring secure communication (usually via SSL), authorization, authentication, permissions etc. The ✓ sign indicates that an agent platform has a particular property meanwhile ✗ sign marks that such a

property is missing. If a **?** sign is used, there is no reference to such a property in available sources and it can be assumed, that the platform does not have it.

As can be seen in Table 1, there are lightweight versions of actually three agent platforms. First two of them, both open source, are *LEAP – Lightweight Extensible Agent Platform* (`http://leap.crm-paris.com`) as a J2ME version of JADE and *MicroFIPA-OS* (`http://fipa-os.sourceforge.net`) as a lightweight version of FIPA-OS. There should be a lightweight version of GrassHopper called *GrassHopper MicroEdition* available at the GrassHopper website (see [5]), but currently there is no sign of such an edition there. At the Tryllian web pages (`http://www.tryllian.com/adk-product/new-body_products-FeaturesBenefits.shtml`) a lightweight ADK runtime environment based on Java2 Standard Edition is mentioned, but it does not seem to be a different ADK version targeted to small-footprint devices.

Regarding the FIPA compatibility, only JADE, FIPA-OS, ZEUS and Comtec Agent Platform (this one unfortunately comes with a poor English documentation) have full FIPA compliancy embedded in the core. The GrassHopper platform becomes FIPA compliant only with a special plug-in – *FIPA Addon* provided as the open source under GNU General Public License (the license of the GrassHopper itself does not allow to use it for any commercial purposes). Also the JACK agent platform needs a plug-in called *FIPA-JACK* to become FIPA compliant. This add-on is being developed at the RMIT University in Melbourne, Australia and can be freely downloaded from `http://www.cs.rmit.edu.au/agents/protocols` website. The ADK is partially FIPA compliant – it implements a subset of FIPA message performatives but support for neither agent management nor message transport protocol has been implemented.

The Java Agent Services (JAS) have also been included in Table 1. However, this project is aimed at the development of the standard JAVA APIs (under the `javax.agent` namespace), i.e. a set of classes and interfaces for the development of your own FIPA-compliant agent-based systems. From this perspective, JAS cannot be considered as a classical agent platform, since it doesn't provide any runtime environment that could be used to run your agents (either on a PC or possibly on an automation controller).

The JINI technology [10] has not been considered in this evaluation either. Similarly to JAS, JINI is a set of APIs and network protocols (based on JAVA Remote Method Invocation) that can help you to build and deploy distributed systems. It is based on the idea of *services* providing useful functions on the network and the *lookup service* that helps clients to locate these services. Although JINI provides a solid framework for various agent implementations (see e.g. [1]), it cannot be regarded itself as an agent platform.

3 Message Sending Speed Benchmarks

It has been discussed earlier, that a speed at which the messages are exchanged among agents can be a crucial factor in agent-based real-time manufacturing applications. Thus we have put selected agent platforms through a series of tests where the message delivery times have been observed under different conditions.

Table 1. Agent platforms overview

JAVA-based Agent Development Toolkits/Platforms - Overview							
Agent platform	Developer	FIPA compatibility			Open-source	J2ME version (lightweight)	Security (authent., SSL, ...)
		Agent Communication (ACL, protocols, ...)	Agent Management (AMS, DF, ACC)	Inter-platform Messaging (MTP)			
JADE (Java Agent DEvelopment Framework)	CSELT http://jade.cselt.it	✓	✓	✓	✓	✓	✓
FIPA-OS	Emorphia http://fipa-os.sourceforge.net	✓	✓	✓	✓	✓	✓
ZEUS	British Telecom www.labs.bt.com/projects/agents/zeus	✓	✓	✓	✓	✗	✓
JACK (Jack Intelligent Agents)	Agent Oriented Software http://www.agent-software.com	✓	✓	✓	✗	✗	✓
GRASSHOPPER 2	IKV++ Technologies AG http://www.grasshopper.de	✓	✓	✓	✗	✓	✓
ADK (Agent Development Kit)	Tryllian http://www.tryllian.com	✓	✗	✗	✗	?	✓
JAS (Java Agent Services API)	Fujitsu, HP, IBM, SUN, ... http://java-agent.org	✓	✓	✓	✓	✗	?
AgentBuilder	IntelliOne Technologies http://www.agentbuilder.com	✗	✗	✗	✗	✗	?
MadKit (Multi-Agent Development Kit)	MadKit Team http://www.madkit.org	✗	✗	✗	✓	✗	?
Comtec Agent Platform	Communication Technologies http://las.comtec.co.jp/ap	✓	✓	✓	✓	✗	✗
Bee-gent	Toshiba [illegible]	✗	✗	✗	✗	✗	✓
Aglets	IBM Japan http://trl.ibm.com/aglets	✗	✗	✗	✓	✗	✓

In each test, so called *average roundtrip time* (avgRTT) is measured. This is the time period needed for a pair of agents (let say `A` and `B`) to send a message (from `A` to `B`) and get a reply (from `B` to `A`). We use a JAVA `System.currentTimeMillis()` method which returns the current time as the number of milliseconds since midnight, January 1, 1970. The roundtrip time is computed by the `A`-agent when a replay from `B` is received as a difference between the receive time and the send time. An issue is that a millisecond precision cannot be mostly reached; the time-grain is mostly 10ms or 15ms (depending on the hardware configuration and the operating system). However it can easily be solved by repeating a message exchange several times (1000 times in our testing) and computing the average from all the trials.

As can bee seen in Table 2, three different numbers of agent pairs have been considered: 1 agent pair ($A-B$) with 1000 messages exchanged, 10 agent pairs (A_1-B_1, A_2-B_2, …, $A_{10}-B_{10}$) with 100 messages exchanged within each pair and finally 100 agent pairs (A_1-B_1, A_2-B_2, …, $A_{100}-B_{100}$) with 10 messages per pair. Moreover, for each of these configurations two different ways of executing the tests are applied. In the *serial* test, the `A` agent from each pair sends one message to its `B` counterpart and when a replay is received, the roundtrip time for this trial is computed. It is repeated in the same manner N-times (N is 1000/100/10 according to number of agents) and after the N-th roundtrip is finished, the average response time is computed from all the trials. The *parallel* test differs in such a way that the `A` agent from each pair sends all N messages to `B` at once and then waits until all N replies from `B` are received. In both the cases, when all the agent pairs are finished, from their results the total average roundtrip time is computed.

As the agent based systems are distributed in their nature, all the agent platforms provide the possibility to distribute agents on several computers (hosts) as well as run agents on several agent platforms (or parts of the same platform) within one computer. Thus for each platform, three different configurations have been considered: (i) all agents running on one host within one agent platform, (ii) agents running on one host but within two agent platforms (i.e. within two Java Virtual Machines – JVM) and (iii) agents distributed on two hosts. The distribution in last two cases was obviously done by separation of the `A-B` agent pairs.

The overall benchmark results are presented in Table 2. Remind that the results for *serial* tests are in milliseconds [ms] while for *parallel* testing seconds [s] have been used. Different protocols used by agent platforms for the inter-platform communication are also mentioned: Java RMI (Remote Method Invocation) for JADE and FIPA-OS, TCP/IP for ZEUS and UDP for JACK.

To give some technical details, two Pentium II processor based computers running 600 MHz (256 MB memory) with Windows 2000 and Java2 SDK v1.4.1_01 were used.

Some of the tests, especially in the case of 100 agents, were not successfully completed mainly because of communication errors or errors connected with the creation of agents. These cases (particularly for FIPA-OS and ZEUS platforms) are marked by a ✖ symbol.

Table 2. Message delivery time results for selected agent platforms.

JAVA-based Agent Development Toolkits/Platforms - Benchmark Results						
Agent Platform	**Message sending - average roundtrip time (RTT)**					
	agents: 1 pair messages: 1.000 x Δ		agents: 10 pairs messages: 100 x Δ		agents: 100 pairs messages: 10 x Δ	
	serial [ms]	parallel [s]	serial [ms]	parallel [s]	serial [ms]	parallel [s]
JADE 2 5	0.4	0.36	4.4	0.22	57.8	0.21
JADE v2.5 1 host, 2 JVM, RMI	8.8	4.30	85.7	4.34	1 426.5	4.82
JADE v2.5 2 hosts, RMI	6.1	3.16	56.2	3.60	939.7	3.93
FIPA-OS 2	28.6	14.30	607.1	30.52	2 533.9	19.50
FIPA-OS 2 s 2 V	20.3	39.51	205.2	12.50	×	×
FIPA-OS 2 2 hosts, RMI	12.2	5.14	96.2	5.36	×	×
ZEUS 4	101.0	50.67	224.8	13.28	×	×
ZEUS v1.04 1 host, 2 JVM, ?	101.7	51.80	227.9	×	×	×
ZEUS v1.04 2 hosts, TCP/IP	101.1	50.35	107.6	8.75	×	×
JACK 5	2.1	1.33	21.7	1.60	221.9	1.60
JACK 5 s 2 V U	3.7	2.64	31.4	3.65	185.2	2.24
JACK v3.51 2 hosts, UDP	2.5	1.46	17.6	1.28	165.0	1.28
NO NAME	141.3	1.98	2 209.3	0.47	×	×
NO NAME 1 host, 2 JVM, ?	N/A	N/A	N/A	N/A	N/A	N/A
NO NAME 2 hosts, IIOP	158.9	×	×	×	×	×

More transparent representation of these results in the form of bar charts is depicted in Figure 2. The left hand side picture in each row corresponds to the serial test with one agent pair while the right hand side picture corresponds to the serial test with ten agent pairs. The first row represents tests run on one host, the second row corresponds to test run also on one host but within two JVMs and the third row shows results of testing on two hosts.

4 Conclusion

Evaluation presented in this paper gives an overview of what agent development tools are currently available and how do they fit the specific requirements of manufacturing control applications.

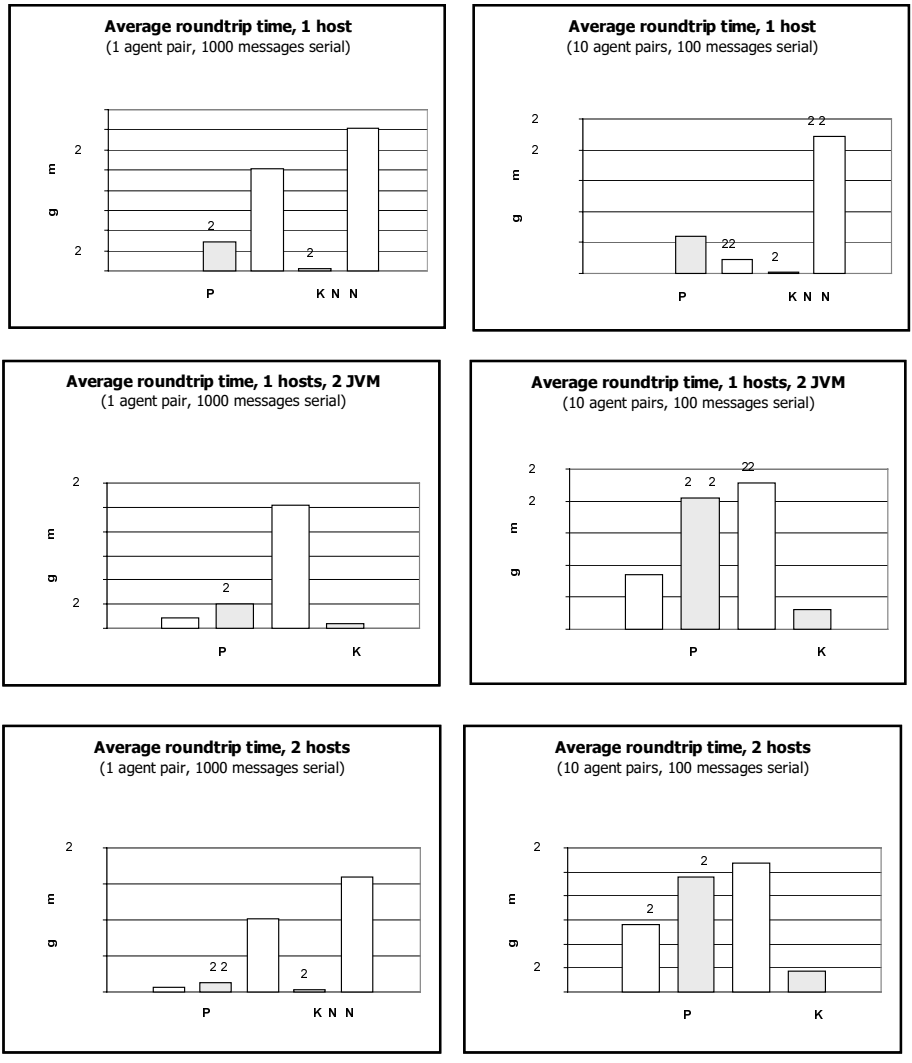


Fig. 2. Serial test results: 1st row – one host, 2nd row – one host with two agent platforms, 3rd row – two hosts. Left hand side – 1 agent pair, right hand side – 10 agent pairs.

On the basis of results of this study, the JADE agent platform seems to be the most suitable open-source candidate for the development tool and the runtime environment for agent-based manufacturing solutions. In comparison with its main competitor, FIPA-OS, the JADE platform offers approximately twice the speed in message sending and, above all, much more stable environment especially in the case of larger number of agents deployed. Also the memory utilization is better optimised in JADE – one agent takes approximately 5.1 KB compared to 14.4 KB in FIPA-OS (see

Figure 3). Another issue in FIPA-OS that we have discovered is the steadily increasing consumption of a computer memory that occurs during the ongoing inter-agent communication in which the agents do not utilize any of the implemented FIPA interaction protocols [6] and just send simple `inform` messages to each other. It seems that even if an agent does not plan to use the interaction protocol (just informs the other agent about something without awaiting any replay), the sender agent automatically creates a conversation object when dispatching a message as well as the receiver agent when a message is received. It means that for each message either sent or received there are such objects held in the internal agent memory. This inevitably leads to an "insufficient memory" error generated by JAVA after some time. The only non-standard solution possible is to manually remove the conversation objects for each message, what has been allowed in FIPA-OS since the version 2.0.

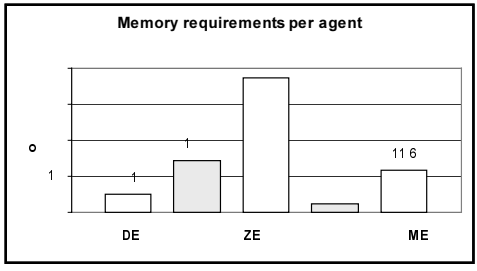


Fig 3. Approximate memory requirements per agent. Memory used by JAVA computed as `java.lang.Runtime.totalMemory() - java.lang.Runtime.freeMemory()`. Results obtained as difference between the memory used before and after the creation of 100 agents.

The ZEUS agent platform shall be unfortunately denoted as the less suitable not only because of its worse performance and stability. We found a way of agent classes definition in ZEUS to be a major difficulty. At the first sight there is a powerful graphical environment where the agents, their tasks and relationships can be defined and then JAVA source code automatically generated on a single mouse click. However, the problem is that for each agent specified in the GUI, there is its own JAVA class file containing the agent's name generated. Moreover, this class file is in a form of a standalone JAVA program, where the agent of that fixed name is created in the `main()` method. It means, that you cannot treat it as a definition of an agent class and use in a standard way to create as many agent instances with different names (and other attributes) as needed. It considerably embarrasses the development of agent applications in ZEUS following the object oriented conventions and as a result fails in majority of agent systems where the number of agents and their names are not known in advance and dynamically change at runtime as new agents join the community (are e.g. generated on a user request or by other agents), migrate between communities, clone themselves etc.

Among the commercial agent platforms, to-date only JACK can offer full FIPA compliancy and also cross-platform interoperability through a special plug-in – JACK agents can send messages e.g. to JADE or FIPA-OS agents (and vice versa) via the FIPA message transport protocol based on HTTP. In the intra-platform

communication, based on UDP protocol, JACK (unlike other platforms) keeps pace with JADE in case of one host and even surpasses it in other cases being approximately 2-3 times faster. Considering the full implementation of the BDI (Belief-Desire-Intention) model, JACK can be regarded as a good alternative to both the open source JADE and FIPA-OS platforms.

References

1. Ashri, R., Luck, M.: Paradigma: Agent Implementation through Jini. In: *Proceedings of the Eleventh International Workshop on Database and Expert Systems Applications*, eds. Tjoa, A.M., Wagner, R. R., Al-Zobaidie, A., IEEE Computer Society, (2000), pp. 53–457
2. Berger, M., Rusitschka, S., Toropov, D., Watzke, M., Schlichte, M.: Porting Distributed Agent-Middleware to Small Mobile Devices. In: *Proceedings of the Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices*, Bologna, Italy, (2002)
3. Deen, S.M., *Agent-Based Manufacturing: Advences in the Holonic Approach*, Springer-Verlag Berlin Heidelberg, (2003)
4. FIPA organization: FIPA ACL Message Structure Specification. document identifier SC00061, http://www.fipa.org/specs/fipa00061
5. FIPA organization: FIPA Inform! Newsletter. **2**(1), (2001), website: http://www.fipa.org/docs/output/f-out-00087
6. FIPA organization: FIPA Interaction Protocols Specifications: website http://www.fipa.org/repository/ips.php3
7. Fletcher, M., Brennan, R.W.: Designing a holonic control system with IEC 61499 function blocks. In: *Proceedings of the International Conference on Intelligent Modeling and Control*, Las Vegas, (2001)
8. Foundation for Intelligent Physical Agents: website http://www.fipa.org
9. Java2 Platform, Micro Edition: website http://java.sun.com/j2me
10. JINI technology: website http://www.jini.org
11. Koestler, A., *The Ghost in the Machine*, Arkana Books, London, UK, (1967)
12. Marik, V., Fletcher, M., Pechoucek, M.: Holons & Agents: Recent Developments and Mutual Impacts. In: *Multi-Agent Systems and Applications II*, LNAI No. 2322, Springer-Verlag, Heidelberg, (2002), pp. 233–267
13. Marik, V., Pechoucek, M., Vrba, P., Hrdonka, V.: FIPA Standards and Holonic Manufacturing. In: *Agent Based Manufacturing: Advances in the Holonic Approach*, ed. Deen, S. M., Springer-Verlag Berlin Heidelberg, (2003), pp. 89–121
14. McFarlane, D.C., Kollingbaum, M., Matson, J., Valckenaers, P.: Development of algorithms for agent-based control of manufacturing flow shops. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Tucson, (2001)
15. Searle, J.R., *Speech Acts*, Cambridge University Press, (1969)
16. Van Leeuwen, E.H., Norrie, D.: Intelligent manufacturing: holons and holarchies. *Manufacturing Engineer*, **76**(2), 86–88, (1997)

An Approach to the Formal Specification of Holonic Control Systems

Paulo Leitão[1], Armando W. Colombo[2], and Francisco Restivo[3]

[1] Polytechnic Institute of Bragança, Quinta Santa Apolónia, Apartado 134, P-5301-857 Bragança, Portugal,
pleitao@ipb.pt
[2] Schneider Electric GmbH.
Steinheimer Str. 117, 63500 Seligenstadt, Germany,
armando.colombo@modicon.com
[3] Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, P-4200-465 Porto, Portugal,
fjr@fe.up.pt

Abstract. In the manufacturing world, globalisation leads to a trend towards the reduction of batches and product life cycle, and the increase of part diversity, which are in conflict with other requirements, such as the cost reduction achieved with higher productivity. Thus, the challenge is to develop flexible, agile and intelligent management and control architectures that satisfy the referred requirements. The holonic manufacturing and the agent-based manufacturing approaches allow a new approach to the manufacturing problem, through concepts such as modularity, decentralisation, autonomy and re-use of control software components. ADACOR, one of the holonic architectures recently proposed, defines a set of autonomous and intelligent holons aiming to improve the performance of control system in industrial scenarios characterised by the frequent occurrence of unexpected disturbances. The formal modeling and validation of the specifications of the ADACOR-holons and of the interactions between these holons to implement the manufacturing control functions is of critical importance. In this paper, a formal methodology is introduced and applied to model the dynamic behaviour of the ADACOR-holon classes.

1 Introduction

The emergent requirements of global markets are leading the manufacturing world to the reduction of batches and product life cycle, and the increase of part diversity, which are in conflict with other important requirements, such as the reduction of costs, achieved normally with higher productivity. A Flexible Manufacturing System (FMS) aims to fill the gap between the mass production, which guarantees high productivity, and the dedicated NC machine production, which guarantees high flexibility and customisation. A FMS is a production structure that comprises a set of workstations, such as machine tools, storage facilities,

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 59–70, 2003.

and robots, interconnected by a transport and materials handling system, and controlled by a supervisory control system.

The flexibility and performance of FMSs depend not only of the individual components, i.e. workstations, storage facilities, etc., but also of the flexibility and performance of the embedded control system [1]. The manufacturing control systems are concerned with coordinating the manufacturing resources to make the desired products. The main functions presented in the manufacturing control system can be split into: process related functions (process planning) and resource allocation related functions. The resource allocation comprises the following main functions: resource allocation planning (scheduling), resource allocation plan execution (dispatching, monitoring, diagnosis, reaction to disturbances, etc.) and pathological state handling (deadlock handling, etc.). The scheduling determines an optimal route with respect to some performance criteria, and the plan execution performs the final assignment of resources to the orders, based on the actual state of the manufacturing system and the schedule plans. The pathological state handling intends to keep the system in a safe state and/or recovers it from undesirable states.

The manufacturing systems are typically heterogeneous environments, comprising heterogeneous hardware components and software applications, with distributed functions, knowledge and skills, which are required to cooperate in order to achieve common goals. The control system should be therefore be based in distributed and autonomous entities, expandable, being possible the addition of new components without the need of re-design, re-programming and re-initialisation of the other components, and re-configurable, adapting dynamically to configuration changes, without stopping or re-starting the process. Additionally, the manufacturing systems are complex non-linear systems, since the occurrence of a disturbance causes non-linear impact in the system. For this reason, their occurrence may have severe impact in the performance of manufacturing systems, being also necessary to improve the system performance in terms of response to change. All these necessary facilities lead to the concept of agile manufacturing systems.

The challenge is to develop new flexible, agile and intelligent management and control architectures that address the above referred problems and requirements. The holonic manufacturing and the agent-based manufacturing approaches that have been introduced in the manufacturing domain by several research teams, such as referred in [2,3,4,5,6,7], allow a new approach to the manufacturing problem, through the concepts of modularity, decentralization, autonomy and re-use of control software components. One of the holonic architectures proposed during the last two years is the ADACOR (Adaptive Holonic Control Architecture for Distributed Manufacturing Systems) architecture [8], which defines a set of autonomous, self-organised and intelligent holons in order to improve the performance of control system in industrial stochastic scenarios, characterised by the frequent occurrence of unexpected disturbances.

The formal modeling and validation of the structural and behavioural specifications of the ADACOR-holons and the interactions between these holons to

implement the manufacturing control functions assumes critical importance. In this paper, a formal methodology to model the behaviour of the ADACOR-holon classes is discussed and applied.

This paper is organised as follows: First, Section 2 discusses the need for a formal methodology to model the specifications of holonic control systems. In Section 3 it is described the specifications and modelled the behaviour of ADACOR-holon classes using Petri Nets (PN) modelling tool. Finally, Section 4 rounds up the paper with conclusions and an overview of planned further developments related to the approach presented here.

2 Formal Methodology

In order to formalise the structure and the behaviour of the holonic manufacturing control systems, and to validate its behaviours and particularity to analyse the co-operation and interaction between the distributed holons, aiming to understand and synthesise the structure and behaviour of the system, it is important to count with a formal modelling methodology.

The proposed methodology for the formal modelling of holonic applications, as illustrated in Fig. 1, combines the UML (Unified Modelling Language) and the PN modelling tools. UML [9] is an object oriented based modelling tool, that is adequate to model the structure and the static aspects of a manufacturing system. In the proposed formal methodology, the static aspects are modelled using mainly the class diagrams, which shows the classes of objects in the system, the attributes and methods for each class, and the relationships between the objects.

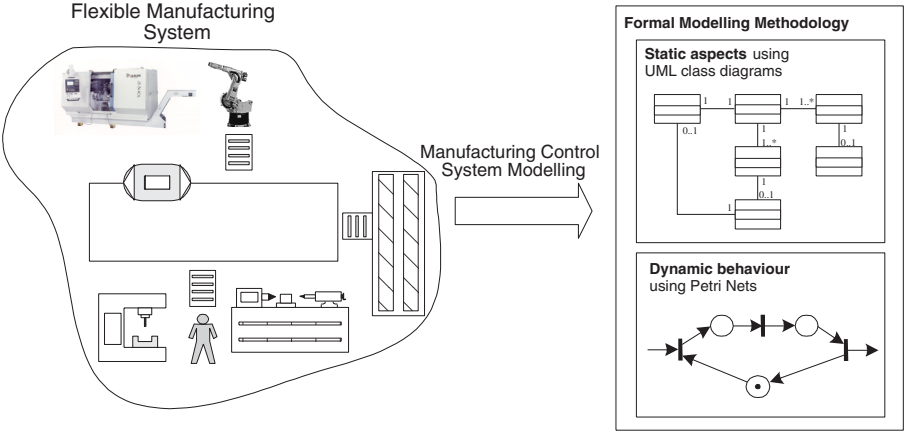


Fig. 1. Modeling a Manufacturing Control System

The modelling of the dynamic behaviour of the system requires a formal tool that captures characteristics like concurrency or parallelism, asynchronous operations, deadlocks, conflicts or resource sharing, which are inherent to FMS [10]. Additionally, it is crucial that the formal modelling tool has the capability to validate the behavioral characteristics of these event-driven systems, as also the analysis of other important aspects, such as the deadlock detection and the performance analysis.

The UML modelling tool doesn't support efficiently the modeling of the dynamic behaviour aspects and the formal validation of these specifications. On the other hand, the PNs is a formal modeling tool, both graphical and mathematical, that seems adequate to model and analyse the structure and the dynamic behaviour of complex event-driven systems with high distribution degree. In comparision with UML, the PN formalism allows to design the control system behaviour, but also to validate and verify the behaviour of the system, based in mathematical background embedded in the PN formalism. In this sense, the proposed methodology uses the PN formalism to model the dynamic behaviour of the holonic manufacturing control system. More details about PN theory and mathematical fundamentals are out of the scope of this work. We recommend the readers to consult the following references [4,12,13].

In industrial manufacturing applications, the PN models become highly complex and difficult to handle. This leads to the definition and application of different types of High-Level Petri Nets, depending on the area of application, i.e., modelling and qualitative analysis, quantitative/performance analysis, modelling of big complex manufacturing environments, supervisory control code generation, etc. [1,14,15].

In the following, a kind of PN tailored for production management and control modelling purposes, proposed in [12], will be used to model the dynamic behaviour of the different holon-types defined in ADACOR holonic architecture, allowing to get a comprehensive formal view of the structure and behaviour of these holon-types.

3 Modelling the Holons Dynamic Behaviour in ADACOR Architecture

The ADACOR architecture proposes a holonic approach to introduce the dynamic and agile adaptation to disturbances in flexible manufacturing systems [8]. Aiming to support the distribution of skills and knowledge, the architecture is based on a set of autonomous, intelligent and co-operative entities, designated by holons, to represent the factory components. These distributed components can be both physical resources (numerical control machines, robots, programmable controllers, etc.) and logic entities (products, orders, etc.). According to the generalization concept of the object-oriented paradigm, the ADACOR architecture groups the manufacturing holons into product, task, operational and supervisor holon classes [16].

The dynamics and behaviour evolution of a manufacturing control system will be modelled through the modeling of dynamic behaviour of each individual holon class in the system using PNs, with the places representing the state of the holons when executing activities and the transitions representing the trigger of actions and the synchronization between holons or between threads within a holon. The tokens in those PN models can represent resource states, parts in the system or logical control.

3.1 Product Holon Model

Each product is represented by a *product holon* that contains all knowledge related to the product and is responsible for the process planning. The product holon receives orders to execute products, which can be customer orders from customer entities or forecast orders based in historic information in case of production to stock. To execute the products, the product holon require the information about the BOM (Bill of Materials) that specifies the product structure, and the process plan required to execute the product, that should be provided in the product data model, created by the engineering department during the product design phase.

In functional terms, which PN-model is illustrated in Fig. 2, the product holon start its execution entering in a state waiting for new product orders. These new orders will generate a new thread to handle the execution of each order, continuing the product holon waiting for new orders, being able to process simultaneously several product orders.

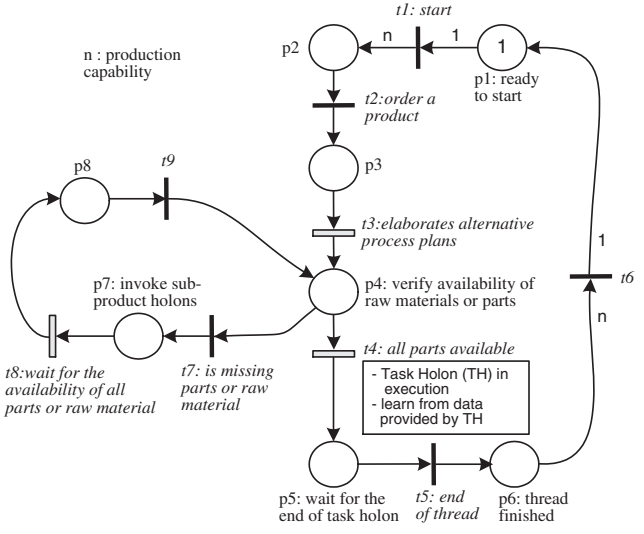


Fig. 2. Product Holon Behaviour Model

In each thread, initially it is elaborated several alternative plans for the execution of the product, based in the knowledge related to the operations routing and in the available resources in the system, indicating for each operation a set of alternative resources to execute it, sorted by the confidence degree on concrete resource to execute it. Then it is verified if the required parts or raw material are available on the storage system, interacting with the operational holon responsible for the storage management. If the raw materials or parts are not available, the product holon interact with other product holons to request the execution of sub-products, according the product structure. When all sub-products or raw materials are available it launches a task holon that will be responsible for the supervision of the manufacturing order leading to the product execution.

After launching a new task holon to deal with the execution of the manufacturing order, the thread will wait for the conclusion of the manufacturing order. At this moment, the task holon notifies the product holon providing the relevant information concerning the execution of the part (for example the process plan used, the start and end dates, etc.). These data should be carefully analysed, allowing from the execution of the manufacturing order, to learn to elaborate more efficient and accurate process plans. The process plans generated in the future will take into consideration penalties to the resources that in previous operations had failures, delays or operations with low quality, and rewards to the resources that executed with success previous operations.

3.2 Task Holon Model

Each available manufacturing order is represented by a *task holon*, which is responsible for the control and supervision of the manufacturing order execution and contains the dynamic information. The task holon functions comprise the order decomposition, resource allocation planning and resource allocation plan execution, as illustrated in the PN-model of Fig 3.

Initially, the task holon requests a pallet and material to the transport and storage system. According to the production type, the pallet may can contain several parts of the same material or contain all necessary parts to execute the final part.

In the resource allocation process, the task holon announces the operations belonging to the manufacturing order, by interacting with available supervisor and operational holons, deciding the allocation of each operation according the bids received.

After allocating all operations, the task holon will start the execution of operations, interacting with the operational holons, even if the allocation is provided by the supervisor holons. For each operation it is necessary to prepare the operation to be executed. This preparation involves the transportation of the parts to the machine where the parts will be processed, and the execution of set-ups in the machine, if necessary. Before requesting the execution of transport of the part to the machine, the task holon should ask if the machine can accept to receive the part that will be transported (for example, to avoid deadlocks due to no space in the machine buffer).

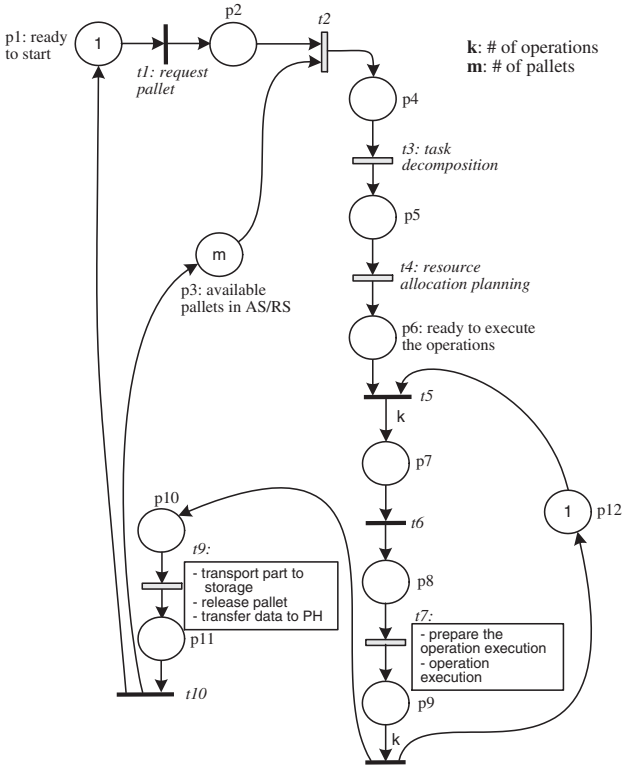


Fig. 3. Task Holon Behaviour Model

When both processes, the transportation of the part and the execution of the set-up, are completed, the task holon can start the execution of the operation, notifying the operational holon. Once the operation is started, the control is given to the operational holon, and the task holon waits for the end of the operation. The described procedure for one operation is repeated for all operations that belong to the process plan of the manufacturing order.

After the execution of all operations, the task holon request the transportation of the part to the storage system and releases the pallet. The task holon finishes its execution transferring to the product holon, the relevant information about the product execution, such as the start and end dates.

3.3 Operational Holon Model

The *operational holons* represent the physical manufacturing resources, such as operators, robots and numerical control machines, managing its behaviour according the resource goals, constraints and skills, and optimising its schedule agenda, Fig. 4.

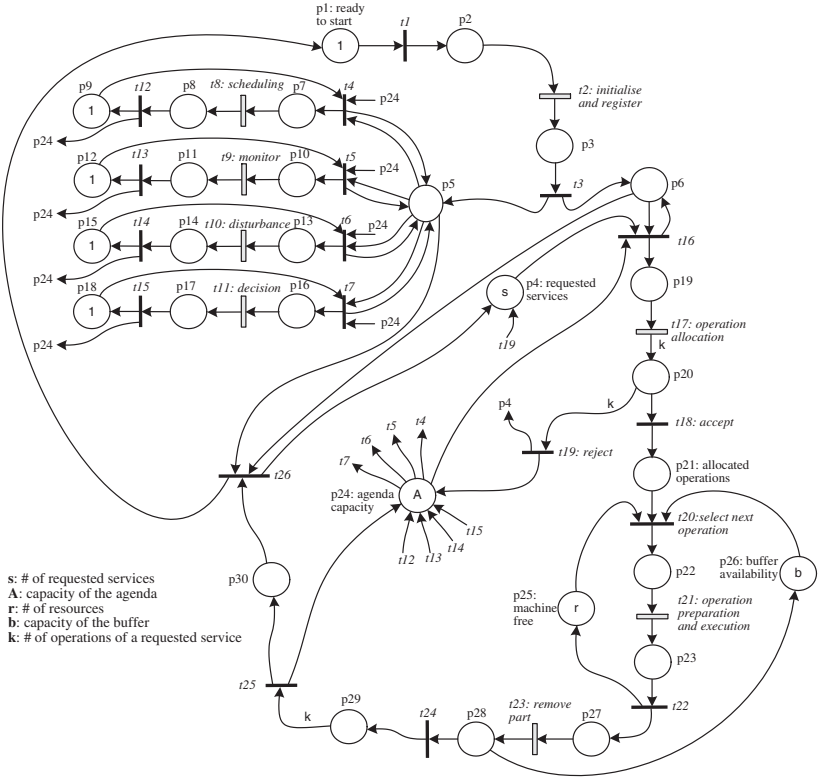


Fig. 4. Operational Holon Behaviour Model

After initialising its components and registering its capacity and skills in the appropriated supervisor holon, according to the organisational structure, the operational holon behaviour acts as a reactive server, in the sense that it is waiting for new operations (proposed by the supervisor holon or by the task holons) and it has the possibility to execute monitoring, scheduling and disturbances handling.

The sub-behaviours are handled asynchronously using threads, so that the execution of one process doesn't block the execution of another process; for example, when monitoring the execution of an operation, the operational holon can handle the announcement of new operations or execute scheduling. The monitoring, disturbance handling and scheduling activities are performed concurrently.

The operation allocation is analysed, the acceptance of the operation allocation being decided according the autonomy factor and the actual agenda capacity. In case of acceptance, the operation is stored in the agenda, waiting for the appropriate moment to start the operation execution. According to the availability of the buffer and the state of the machine, the next operation is selected, based in the local scheduling.

The preparation of the execution involves the transportation of the part to the machine and the execution of a set-up if necessary. The set-up aims to endow the machine with the required tools and fixtures to execute the operation, and in case of need to execute a set-up, the operational holon deals directly with the operational holon that represents the team that executes the set-ups. After this phase, it is started the execution of the operation.

When the operation finishes, the resource returns to the idle state being able to initiate the execution of another operation, and the part is removed from the machine to the next machine, according to the resource allocation plan.

3.4 Supervisor Holon Model

The product, task and operational holons are quite similar to the product, order and resource holons, presented at the PROSA reference architecture [2] . The *supervision holon* presents different characteristics from the staff holons defined in PROSA, introducing coordination and global optimisation in decentralised control approaches, coordinating several operational and supervisor holons. In normal operation, the supervisor holon coordinates the activity of the holons under its domain, while when a disturbance occurs, these holons may have to find their way without the help of the supervisor holon.

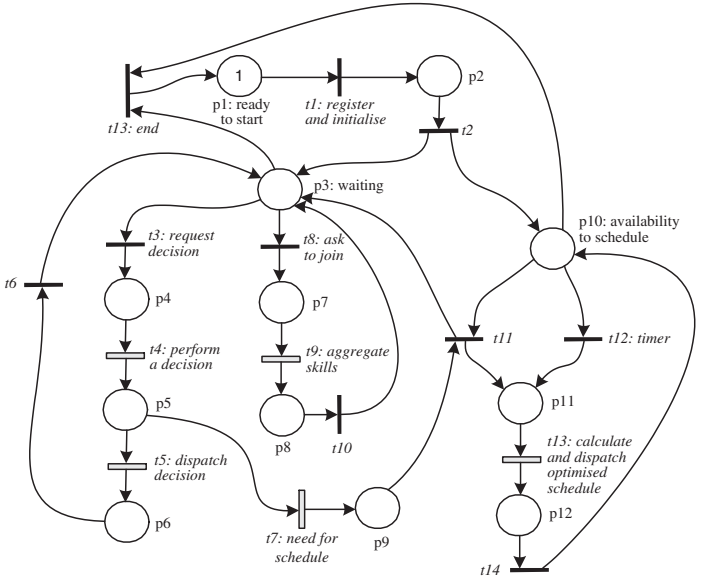


Fig. 5. Supervisor Holon Behaviour Model

The supervisor holon is also responsible for the group formation and their dynamic evolution according the environment context, based in pre-defined clus-

ters of holons, combining synergies, aggregating skills and offering the combined services to external entities in the manufacturing system. These groups can be formed to build a shop floor, a manufacturing cell, or a machine equipped with a set of tools, assuming the supervisor holon the role to coordinate each group.

The behaviour model of the supervisor holon acts simultaneously as a server and a client: as a server waiting for requests and as a client generating optimised schedules that are sent to the operational holons, as illustrated in Fig. 5.

The request to elaborate optimised schedules can be generated from the announcement of an operation, from the need to optimise the actual schedule after the end or disruption of an operation, or from the need for re-scheduling due to a disturbance. After achieving a schedule, the new scheduled operations, and the allocated operations that had modifications in schedule parameters due to the new schedule, are proposed to the appropriate operational holons and to the task holons.

The decision-making activity is related to the actions associated to monitoring, scheduling and disturbance handling, presenting more complexity than to the one presented at operational holons due to the need to handle with aggregated lower holons knowledge and skills. As the supervisor holon co-ordinates several operational and/or supervisor holons, it manages the group of holons under its coordination domain, aggregating the skills and capacity of the operational holons, when they join to the group.

4 Conclusions

The flexible manufacturing systems are complex and stochastic environments requiring the development of flexible, agile and intelligent management and control architectures that support the small batches, product diversity, high quality and low costs imposed by global markets. One of these intelligent management and control architectures is ADACOR.

The ADACOR holonic architecture aims to improve the performance of control system in scenarios characterised by the frequent occurrence of unexpected disturbances, defining four main holon classes: product, task, operational and supervisor holons. Each of these holons presents characteristics of autonomy, cooperation and intelligence, allowing to implement an intelligent distributed control system.

In this paper, it is used a kind of Petri net, tailored for production management and control modelling purposes, as a formal methodology to model the behaviour of the ADACOR-holon classes in a bottom-up approach. The individual model of each holon uses special temporised transitions to model activities execution, that can be exploded into a more detailed and refined level. These sub-models, according the degree of refinement, are the different software control modules of the hardware, i.e., a formal representation of the holons.

The edition, simulation, qualitative and quantitative (performance) analysis, and formal validation of the structural and behavioural specifications of the

ADACOR-holons and their interactions, is one of the complementary works to the approach proposed here and it can be found in [17].

A brief overview of the latest works published in the area (see e.g. [18,19]) allows to identify a set of weak points in using the PN-formalism proposed here and other similar extensions of this tool. This is particularly true if the system presents many instances of the same component (e.g., n resources need n operational Holons). In this case, the model will be increased (structure and components) in a non-controllable manner. In our opinion, the use of High-Level PN, such as those proposed in [20], allows to reduce this complexity, by compressing the representation of states, actions and events, to overcome the identified limitations and to support more complex and bigger coordination scenarios.

References

1. Colombo, A.W.: Integration of High-Level Petri Net-based Formal Methods for the Supervision of Flexible Production Systems. Tutorial Lecture at the 1st Online Symposium for Electronics Engineers, February 20th (2001).
2. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L. and Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. In: Computers In Industry, 37, (1998) 255–274.
3. Fisher, K.: Agent-Based Design of Holonic Manufacturing Systems. In: Journal of Robotics and Autonomous Systems, Elsevier Science B.V., 27 (1999) 3–13.
4. Maturana, F. and Norrie, D.: Multi-Agent Mediator Architecture for Distributed Manufacturing. In: Journal of Intelligent Manufacturing, 7 (1996) 257–270.
5. Van Dyke Parunak, H., Baker, A. and Clark, S.: The AARIA Agent Architecture: An Example of Requirements-Driven Agent-Based System Design. In: Proceedings of 1st International Conference on Autonomous Agents, (1998) 482–483.
6. Brennan, R., Fletcher, M. and Norrie, D.: An Agent-based Approach to Reconfiguration of Real-Time Distributed Control Systems. In: IEEE Transactions on Robotics and Automation, 18(4), (2002) 444–451.
7. Colombo, A.W., Neubert, R. and Schoop, R.: A Solution to Holonic Control Systems. In: Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation, Sophia/Nice, France, (2001) 489–498.
8. Leitão, P. Restivo, F.: Adaptive Production Control Systems. In: Proceedings of special session on Agent-based Intelligent Automation and Holonic Control Systems of the 28th Annual Conference of the IEEE Industrial Electronics Society, Sevilla, Spain, 5-8 November (2002) 2968–2973.
9. Rumbaugh, J., Jacobson, I. and Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley, (1998).
10. Silva, M. and Valette, R.: Petri Nets and Flexible Manufacturing. In: Advances in Petri Nets, Lectures Note in Computer Science, vol. 424, Springer Verlag, (1989) 374–417.
11. Murata, T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, 77(4), (1989) 541–580.
12. Colombo, A.W. and Carelli, R.: Petri Nets for Designing Manufacturing Systems. In: Computer-Assisted Management and Control of Manufacturing Systems, cap. 11, S.G. Tzafestas (ed.), Springer-Verlag (1997).
13. Desrochers, A. and Al-Jaar, R.: Applications of Petri Nets in Manufacturing Systems-Modeling, Control and Performance Analysis. IEEE Press (1995).

14. Feldmann, K., Schnur, C. and Colombo, A.W.: Modularized, Distributed Real-time Control of Flexible Production Cells, using Petri Nets. In: CEP, 4(8), (1996) 1067–1078.
15. Holloway, L., Krogh, B. and Giua, A.: A Survey of Petri net Methods for Controlled Discrete-Event Systems. In: Discrete-Event Systems: Theory and Applications, 7(2), Kluwer Academics, (1997) 151–190.
16. Leitão, P. and Restivo, F.: Identification of ADACOR Holons for Manufacturing Control. In: Proceedings of 7th IFAC Workshop on Intelligent Manufacturing Systems, Budapest, Hungary, 6–8 April (2003) 109–114.
17. Leitão, P., Colombo, A.W. and Restivo, F.: A Formal Validation Approach for Holonic Control System Specifications. Submitted to 9th IEEE International Conference on Emerging Technologies and Factory Automation, Lisboa, Portugal, (2003).
18. Vyatkin, V., Hanisch, H-M. and Ivanov,G.: Application of Formal Methods for Deep Testing of Controllers in Holonic Systems. In: Proceedings of the 1st IEEE International Conference on Information Technology in Mechatronics (ITM'2001), Istanbul, Turkey, 1-3 October (2001) 53–58.
19. Frey, G., Minas, M. and John, K.: Steuerungsentwurf mit Petrinetzen. SPS-Magazin, Verlag Marburg, 4/5 (2002) 44–47.
20. Colombo, A. W., Neubert, R., and Süssmann, B.: A Colored Petri Net based Approach Towards a Formal Specification of Agent-Controlled Production Systems. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Tunisia, (2002).

Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems

Klaus Fischer, Michael Schillo, and Jörg Siekmann

DFKI GmbH,
Stuhlsatzenhausweg 3,
D–66123 Saarbrücken,
{kuf, schillo, siekmann}@dfki.de

Abstract. With the growing usage of the world-wide ICT networks, agent technologies and multiagent systems are attracting more and more attention, as they perform well in environments that are not necessarily well-structured and benevolent. Looking at the problem solving capacity of multiagent systems, emergent system behaviour is one of the most interesting phenomena, however, there is more to multiagent systems design than the interaction between a number of agents: For an effective system behaviour we need structure and organisation. But the organisation of a multiagent systems is difficult to specify at design time in the face of a changing environment.
This paper presents basic concepts for a theory of holonic multiagent systems to both provide a methodology for the recursive modelling of agent groups, and allow for dynamic reorganisation during runtime.

1 Introduction

A multiagent system (MAS) consists of a collection of individual agents, each of which displays a certain amount of *autonomy* with respect to its actions and perception of a domain. Overall computation is achieved by *autonomous computation* within each agent and by *communication* among the agents. The capability of the whole MAS is an *emergent functionality* that may surpass the capabilities of each individual agent [19,20]. An extremely useful feature in terms of reduction of complexity for the designer of a MAS is that an overall task can be broken down into a variety of specific sub-tasks, each of which can be solved by a specific agentified problem solver.

Jennings notes that "the development of robust and scalable software systems requires autonomous agents that can complete their objectives while situated in a dynamic and uncertain environment, that can engage in rich, high-level social interactions, and that can operate within flexible organisational structures" [9]. Agents acting in organisational structures can encapsulate the complexity of subsystems (simplifying representation and design) and modularise its functionality (providing the basis for rapid development and incremental deployment). Organisations are social structures which have mechanisms of conflict resolution resulting from previously resolved problems or conflicts [7]. They institutionalise anticipated coordination, which is especially useful for medium- and large-scale applications that require limitation of the agents' communication behaviour.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 71–80, 2003.

With this work, we provide some terminology and theory for the realisation of dynamically organised societies of agents. The central concept for our endeavour, which has been iteratively tested, developed, and applied in a series of projects over the course of several years, is the *holonic multiagent system*. According to Arthur Koestler [12], a *holon* is a self-similar or *fractal* structure that is stable and coherent and that consists of several holons as sub-structures. Koestler gives biological examples. For instance a human being consists of organs which in turn consist of cells that can be further decomposed and so on. None of these components can be understood without its sub-components or without the super-component it is part of.

Many distributed problems exhibit an inherent structure and we need to mirror this structure in the structure of the relationship between (agentified) problem solvers. For this purpose in a holonic multiagent systems, an agent that appears as a single entity to the outside world may in fact be composed of many sub-agents and conversely, many sub-agents my decide that it is advantageous to join into the coherent structure of a super-agent and thus act as single entity — just as the swarm of a certain species of fish sometimes takes on the appearance of a (much bigger) fish. We call agents consisting of sub-agents with the same inherent structure *holonic agents*.

Section 2 gives a formal definition of multiagent systems in general. In Section 3, we extend this to a formal definition of holonic multiagent systems building on previous work in this area [5,6,8], and highlight the diversity of groupings (links of varying nature between agents and recursion) that are possible with this concept. Section 4 compares the notion of holonic multiagent systems to holonic manufacturing systems.

2 Abstract Specification of Multiagent Systems

For any software system, it is common practice to distinguish the static specification of the system from its runtime instance. While concepts and theories for the static specification of software systems are reasonably well-understood, concepts and theories for the specification and analysis of the dynamic behaviour of a software system are by far not as sophisticated. This is especially true if we look at MAS, in which self-organisation is an important aspect. This makes MAS different from systems that are designed according to a more traditional software development paradigm.

We assume that there is some infrastructure which supports the agents in the process of self-organisation. For example, the FIPA[1] initiative has established standards for such infrastructures in an open environment. This paper takes a more abstract point of view, which assumes that there is an agent directory service (ADS) which allows the agents to find out how they can contact other agents that currently exist in the system. This means that we require that the ADS provides at least a white pages service, where agents can inquire the addresses of other agents. Other services like yellow pages, i.e. the information on which services are offered by specific agents, may also be provided by the ADS.

[1] See http://www.fipa.org/

However, this and possibly other more general services could also be introduced by other specialised agents of the MAS.

To describe a concrete MAS for a given application domain, we specify a set of prototypical agents. This static description of the MAS is given by $\mathcal{MAS}_{prot} := (\mathcal{A}_{prot}, ADS)$, where

$\mathcal{A}_{prot}$ is the set $\{A^1, \ldots, A^n\}, n \in \mathbb{N}$ of prototypical agents, instances of which can be dynamically introduced into the system. These agents are the potentially available problem solvers, where several instances of a specific prototypical agent can be created.

ADS is a specialised prototypical agent providing an agent directory service.

We assume that instances of this finite set of agent types can be dynamically introduced into the MAS that executes (i.e. works on a specific problem) in a given application domain.

The process of problem solving starts with the initial agent system

$$\mathcal{MAS}_{init} = (\mathcal{A}_{init}, ADS_{init}) \text{ where}$$

$$\mathcal{A}_{init} = \{A^1_1, \ldots, A^1_{k_1}, \ldots A^n_1, \ldots, A^n_{k_n}\}, k_1, \ldots k_n \in \mathbb{N} \text{ and}$$

$$\forall A^i_j \in \mathcal{A}_{init} : A^i \rhd A^i_j \wedge A^i \in \mathcal{A}_{prot}.$$

$A^i \rhd A^i_j$ (read "A^i *is instantiated by* A^i_j") denotes that A^i_j is an instance of the prototypical agent A^i. This means that A^i_j inherits its behaviour and initial knowledge from A^i but may also have additional knowledge (like for example its unique identification which can be used as an address to communicate with A^i_j).

Note that the explicit introduction of ADS_{init} does not necessarily mean that the MAS is closed in the sense that the system engineer is in control of all parts of the system. We can assume that ADS_{init} represents some ADS, which is already available and which has the state of ADS_{init} at the time when the first agent of the part of the system that is under the control of the system engineer is started. Along the same line of reasoning we can assume that some of the agents in $\mathcal{A}_{init}$ were also not designed by the software engineer but represent agents that are available in the open environment. Let us without loss of generality assume that $\mathcal{A}_{open} = \{A^1_1, \ldots A^1_{k_1}, \ldots A^m_1, \ldots A^m_{k_l}\}$ for some $1 \leq m < n$ represents the set of these agents. The specification for the corresponding prototypical agents $A^1, \ldots A^m$ is likely to be incomplete in the sense that the system engineer who designs $\mathcal{A}_{prot}$ only needs to have the information about $A^1, \ldots A^m$. This is actually needed for the rest of the agents in $\mathcal{A}_{prot}$ to use services that are offered by the former set of agents.

From $\mathcal{MAS}_{init}$ the dynamic $\mathcal{MAS}_t$ evolves as

$$\mathcal{MAS}_t = (\mathcal{A}^t, ADS_t) \text{ where}$$

$$\mathcal{A}_t = \{A^{1,t}_1, \ldots, A^{1,t}_{l_1}, \ldots A^{n,t}_1, \ldots, A^{n,t}_{l_n}\}, l_1, \ldots, l_n \in \mathbb{N} \text{ and}$$

$$\forall A^{i,t}_j \in \mathcal{A}_t : A^i \blacktriangleright A^{i,t}_j \wedge A^i \in \mathcal{A}_{prot}.$$

$\blacktriangleright$ (read "*is transformed into*") denotes $\rhd \circ \rightsquigarrow^*$ which means that we have $A^i \rhd A^i_j$ where $A^i_j \in \mathcal{A}_{init}$ and $A^i_j \rightsquigarrow^* A^{i,t}_j$ where $\rightsquigarrow$ denotes the transformation of A^i_j by a single step of computation.

The computation goes on while the agents send and receive messages. New agents may be introduced and some of the active agents may be terminated. Each agent has a unique address, which an agent can make accessible to all other agents by registering with the *ADS* agent. All agents automatically know the identification of the *ADS* agent.

3 Holonic Multiagent Systems

Multiagent systems represent a new problem solving paradigm [1], where the difficult specification at design time of how a problem should be solved, is all well come by the interaction of the individual agents at run-time and the idea is that the solution of a given problem emerges from this interaction. Looking at nature, an ant hive is a well-known intuitive case, which demonstrates emergent problem solving behaviour :It is impossible to explain the overall behaviour of an ant hive just by the behaviour of an individual ant and the removal of even a significant part of the hive and does not necessarily influence the overall behaviour. Though some parts of the hive seem to be more important than others. Although interesting results have been presented using this approach to problem solving, emergent problem solving behaviour has also been criticised to provide inefficient or even undesirable results.

Divide and conquer is a widely accepted problem solving paradigm of computer science. Here, a centralised problem solving entity accepts a task, separates it into sub-tasks and distributes these sub-tasks to decentralised problem solvers. The problem solvers produce solutions for the sub-problems and send these solutions back to the centralised problem solver which integrates the solutions of the sub-problems into an overall solution for the original task. This approach to problem solving is of course much more structured than the pure emergent problem solving paradigm. The contract-net protocol [17] is a widely-accepted problem solving model in based on the divide and conquer model, where the centralised problem solving entity, called the manager for the task, separates the overall task into sub-tasks. The manager uses a bidding procedure (a first price sealed bid auction) to find the most appropriate decentralised problem solver for each of the sub-problems. The integration of the solutions of the sub-problems into an overall solution is again done by the manager. This procedure can be recursively nested, i.e. the decentralised problem solvers can again use the contract net model to find a set of further problem solvers who are able to solve the given sub-sub-task.

3.1 Definition of a Holonic Multiagent System

The concepts of *fractal* and *holonic* system design in manufacturing were proposed to combine top-down hierarchical organisational structure with decentralised control, which takes the bottom-up perspective [18,3]. Although it is

possible to organise holonic structures in a completely decentralised manner, for efficiency reasons it is more effective to use an individual agent to represent a holon. In some cases, one of the already existing agents is selected as the representative of the holon based on a fixed election procedure. In other cases a new agent is explicitly introduced to represent the holon during its lifetime. Representatives are called the *head* of the holon, the other agents in the holon are called *body*. In both cases, the representative agent represents the shared intentions of the holon and negotiates these intentions with the agents in the holon's environment as well as with the agents internal to the holon. Only the head communicates with the outside of the holon. The binding force that keeps head and body in a holon together can be seen as commitments [16].

Using the formalisation of Section 2, the set $\mathcal{H}$ of all holons in $\mathcal{MAS}_t$ is defined recursively:

- for each $a \in \mathcal{A}_t$, $h = (\{a\}, \{a\}, \emptyset) \in \mathcal{H}$, i.e. every instantiated agent constitutes an *atomic* holon, and
- $h = (Head, Subholons, C) \in \mathcal{H}$, where $Subholons \in 2^{\mathcal{H}} \backslash \emptyset$ is the set of holons that participate in h, $Head \subseteq Subholons$ is the non-empty set of holons that represent the holon to the environment and are responsible for coordinating the actions inside the holon. $C \subseteq Commitments$ defines the relationship inside the holon and is agreed on by all holons $h' \in Subholons$ at creation of the holon h.

A holon h behaves in its environment like any other agent in $\mathcal{A}_t$. Only at closer inspection it may turn out that h is constructed from a set of agents. As any head of a holon has a unique identification, it is possible to communicate with each holon by just sending messages to their addresses. Given the holon $h = (Head, \{h_1, ..., h_n\}, C)$ we call $h_1, ..., h_n$ the *subholons* of h, and h the *superholon* of $h_1, ..., h_n$. The set $Body = Subholons \backslash Head$ (the complement of $Head$) is the set of subholons that are not allowed to represent holon h. Naturally, holons h' are allowed to engage in several different holons at the same time, as long as this does not contradict the sets of commitments of these superholons. We will now outline a treatment of C, a more detailed coverage of this topic can be found in [15].

3.2 Holonic Organisation

For the implementation of a holonic multiagent system, we need to turn to more fine grained issues concerning the commitments that define the intra-holonic relationship. Let us look at some general possibilities for modelling holonic structures. The following notions differ in the degree of autonomy the subholons have and cover the spectrum from full subholon autonomy to a complete lack of autonomy.

A holon as a set of autonomous agents. At one end of the spectrum is a model which assumes that the subholons are fully autonomous agents with their predefined architecture and the superholon is just a new conceptual entity whose

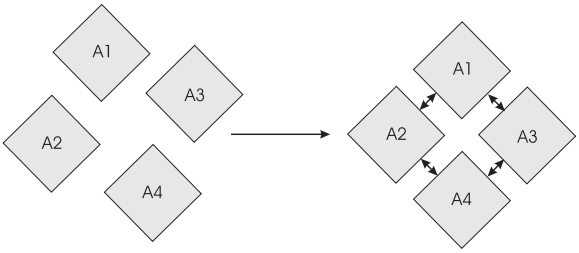


Fig. 1. A holon as a set of autonomous agents.

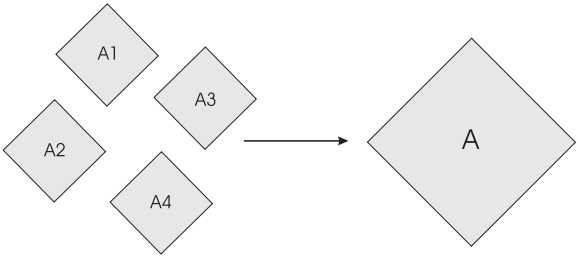


Fig. 2. Several agents merge into one.

properties are made up by the properties of the subholons. Figure 1 displays this constellation. In this case no agent has to give up its autonomy, and the superholon is realised exclusively through cooperation among the subholons. The most transparent way of cooperation for this is an *explicit coordination by commitment* via communication, i.e., agents negotiate over joint plans, task distribution or resource allocation. If commitments can not be established through communication, *implicit coordination* can be achieved in two ways: either, the holons are designed such that a goal directed common behaviour emerges from the behaviour of the sub-agents, or some subholons are able to represent goals and intentions of other agents and reason about them; thus, they coordinate their actions without or at least with little communication.

The representation of a holon as a set of autonomous agents is in a sense just another way of looking at a traditional multiagent system. The holon entity itself is not represented explicitly as a piece of code. In this case, holonic structures are only a design aid for structured agent-oriented programming. This is formally described as holon $h = (\{A_1, A_2, A_3, A_4\}, \{A_1, A_2, A_3, A_4\}, C_{autonomous})$.

Several agents merge into one. The other extreme of the design spectrum terminates the participating sub-agents and creates a new agent as the union of the sub-agents with capabilities that subsume the functionalities of the sub-agents (see Figure 2). In this case the merging agents completely give up their auton-

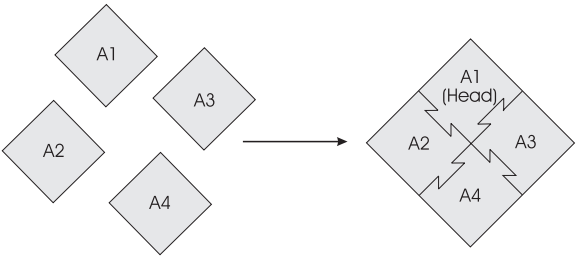


Fig. 3. A holon as a moderated association.

omy but they may be re-invoked when the superholon is terminated. Naturally, this is a new atomic holon $h = (\{A\}, \{A\}, C_{merge})$.

The realisation of this approach assumes procedures for splitting and merging holons that lead to the creation of a new agent. For agents of the same kind with an explicit representation of goals and beliefs (e.g., BDI agents) merging can be achieved by creating an agent with the union of the sub-agents' beliefs and goals provided consistency. Especially for a heterogeneous sets of agents this can be intractable and in either case may not be very desirable.

A holon as a moderated association. The two solutions above are extremes and only useful in very specific circumstances. Hence, we shall propose a continuum, the border lines of which are the two above architectures. Consider a hybrid way of forming a holon, where agents give up only part of their autonomy to the superholon (cf. Figure 3). From a software engineering point of view (in terms of reducing complexity) it is advisable to allow only for a single head which represents the superholon to the rest of the agent population (to reduce coordinational effort). Its competence may range from purely administrative tasks to the authority to give directives to other subholons. Furthermore, the head may have the authority to plan and negotiate for the holon on the basis of its subholons' plans and goals, and even to remove some subholons or to incorporate new subholons. Figure 3 visualises this approach with an example resulting in a holon $h = (\{A_1\}, \{A_1, A_2, A_3, A_4\}, C_{association})$.

Several ways to determine the head are possible. Either, a new agent is created for the lifetime of the holon, or one of the members of the holon takes the role of the head and gains the additional functionality. Or, either one member of the holon is pre-destined for the leadership or an election procedure is needed to promote one of the agents to leadership. Depending on the application domain, the competence of the representative may vary: the resulting structure can range from a loosely moderated association to a authoritative, hierarchical structure. However, the members of the superholon are always represented as agents, and, hence, we do not lose the capability to solve problems in a distributed fashion.

This approach allows for an explicit modeling of holons, a flexible formation of holonic associations, and a scalable degree of autonomy of the participating agents that are subject to negotiation and make up the commitments $C_{association}$

of the superholon (for a more detailed discussion see [15]). The most challenging problem in this design is the distribution of individual and overall computation of the holonic multiagent system.

4 Holonic Multiagent Systems vs. Holonic Manufacturing Systems

Although similar in name, there are several important differences between holonic multiagent systems as proposed here and holonic manufacturing systems as presented in the literature (e.g. [4,14]):

- Modelling the recursion of agent grouping is an integral part of holonic multiagent systems. Mirroring the complex composition of a task, holonic agents can engage in a complex nested structures and nested structures of arbitrary depth are possible and meaningful (depending on the complexity of the task). This is not the case for holonic manufacturing systems.
- Holonic manufacturing systems make no assumption about the internal architecture of the head of a holon, it is only required to act as the control unit. For holonic multiagent systems however, the head is required to possess agent properties (cf. [21]).
- The head of holonic multiagent systems are not required to co-ordinate the work of a physical resource, but instead co-ordinate the work of several information agents that exist only virtually (information agents collaborating for increase of efficiency, to combine competencies or resources, to resolve bottlenecks).
- Holonic manufacturing systems use a market metaphor to design inter-holon co-ordination. Research on holonic multiagent systems is concerned with choosing long-term partners (and is thus related to coalition formation) as well as researching the diversity of possible organisational structures [11].
- In a holonic multiagent system, a holon is not a piece of code. It is merely a concept that is realised by commitments between agents (which exist as code) to maintain a specific relationship concerning goals and has as a result an emergent structure between agents.

While this shows the conceptual differences, it does not rule out the application of holonic multiagent systems to the manufacturing domain, which we have done successfully in several industrial projects.

5 Applications

The proposed theory has been iteratively tested, developed, and applied in a series of projects over several years with a big variation in requirements. In one domain (flexible manufacturing) agents form holons because they have different abilities and can only as a group achieve the task at hand [5]. A second example (train coupling and sharing) demonstrates that even in a setting where we have agents with identical abilities holonic structures can be beneficial [13]. Several

other projects focused on special aspects of holonic modelling (e.g. RoboCup [10], Socionics [15]) The most striking application that used the presented approach to holonic multiagent systems is the TeleTruck system, which was designed to do order dispatching in haulage companies [2].

In this system, the basic transportation units (trucks, trailers, drivers, chassis, and containers) are modeled by agents which temporarily form holons that represent vehicles for the execution of transportation tasks. The vehicle holons are headed by a special agent that is equipped with planning capabilities. All the vehicle holons and the agents representing currently idle transportation units form a super-holon that represents the whole transportation company. The head of the company holon, called the *company agent* coordinates the interaction with the user and communicates with other companies that employ the TeleTruck system. Agents representing transportation units are autonomous in their decision to participate in a vehicle holon. Participating in the holon however restricts the autonomy of the subholons for this time span, since they have to execute the sub-tasks allocated to them. The agents forming a vehicle holon cooperate in order to pursue the goal of executing a set of transportation tasks. Sometimes, even different vehicle holons cooperate for a task. A vehicle holon is able to transport the cargo, which none of its components could do on its own.

6 Conclusion

The paper presents a general framework for holonic multiagent systems, whose advantage is threefold. First, the model preserves compatibility with standard multiagent systems by addressing every holon as an agent, whether this agent represents a set of agents or not. The complexity of a group of agents is encapsulated into a holon represented by its head, the number of agents involved in the holon becomes irrelevant for other agents communicating with it. Secondly, holonic multiagent systems are one way to introduce recursion into the modelling of multiagent systems, which has proven to be a powerful mechanism in software design. Of course a holonic multiagent system is more than just the recursive decomposition into its agents, as we have solved the additional structure preserving problem. Third, there is no restriction to a specific or static association between agents, so it leaves room to introduce a variation of organisational concepts, which can dynamically change at run-time. There is no comparable programming construct that would support the design of such systems in a purely object-oriented programming approach.

References

1. A. H. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.
2. H. Bürkert, K. Fischer, and G. Vierke. Holonic transport scheduling with TELETRUCK. *Applied Artificial Intelligence*, 14(7):697–726, 2000.
3. J. Christensen. Holonic manufacturing systems — initial architecture and standard directions. In *Proc. of the 1st European Conference on Holonic Manufacturing Systems*, Hannover, December 1994.

4. S. M. Deen. A cooperation framework for holonic interactions in manufacturing. In S. M. Deen, editor, *Proceedings of the Second Internatinal Working Conference on Cooperating Knowledge-Based Systems (CKBS-94)*, pages 103–124. DAKE Centre, University of Keele, 1994.
5. K. Fischer. Agent-based design of holonic manufacturing systems. *Journal of Robotics and Autonomous Systems*, 27:3–13, 1999.
6. K. Fischer. Holonic multiagent systems – theory and applications. In *Proceedings of the 9th Portuguese Conference on Progress in Artificial Intelligence (EPIA-99), LNAI Volume 1695*, LNAI. Springer Verlag, 1999.
7. L. Gasser. Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence*, 47:107–138, 1991.
8. C. Gerber, J. Siekmann, and G. Vierke. Flexible autonomy in holonic multiagent systems. In *AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.
9. N.R. Jennings. Agent-based computing: Promise and perils. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1429–1436, 1999.
10. C. G. Jung. Experimenting with layered, resource-adapting agents in the robocup simulation. In *Proc. of the ROBOCUP'98 Workshop*, 1998.
11. T. Knabe, M. Schillo, and K. Fischer. Inter-organizational networks as patterns for self-organizing multiagent systems. In *Proceedings of the Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, In print.
12. A. Koestler. *The Ghost in the Machine.* Hutchinson & Co, London, 1967.
13. J. Lind, K. Fischer, J. Bcker, and B. Zierkler. Transportation scheduling and simulation in a railroad scenario: A multi-agent approach. In *Proc. of the* 4^{th} *int. Conf. on The Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 325–344, London, UK, 1999. The Practical Application Company Ltd.
14. R. Rabelo, L. Camarinha-Matos, and H. Afsarmanesh. Multi-agent perspectives to agile scheduling. In R. Rabelo, L. Camarinha-Matos, and H. Afsarmanesh, editors, *Intelligent Systems for Manufacturing*, pages 51–66. Kluwer Academic Publishers, 1998.
15. M. Schillo. Self-organization and adjustable autonomy: Two sides of the same medal? *Connection Science*, 14(4):345–359, 2003.
16. M. P. Singh. Commitments among autonomous agents in information-rich environments. In *Modelling Autonomous Agents in a Multiagent World*, pages 141–155, 1997.
17. R. G. Smith. The contract net: A formalism for the control of distributed problem solving. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, page 472, 1977.
18. H. Warnecke and M. Hüser. *The Fractal Company — A Revolution in Corporate Culture.* Springer-Verlag, 1995.
19. G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT Press, 1999.
20. M. Wooldridge. *An Introduction to Multiagent Systems.* John Whiley & Sons, 2002.
21. M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

The Link between Autonomy and Organisation in Multiagent Systems

Michael Schillo, Klaus Fischer, and Jörg Siekmann

German Research Center for Artificial Intelligence (DFKI),
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{kuf, schillo, siekmann}@dfki.de

Abstract. Market-based approaches have a long tradition in supporting of task-assignment multiagent systems. Such systems consist of customer agents with jobs to assign, and provider agents that have the resources to perform these jobs. Jobs can be complex in the sense that they require the collaboration of several provider agents. We present a set of organisational forms of collaboration between firms that have the potential to increase performance through the structure they impose. This gain of structure, which comes with a loss of autonomy of the individual agents, is especially valuable in settings where communication has to be limited.

1 Introduction

The central setting for this paper is a market of two kinds of agents, *customers* and *providers*. The customer agents have jobs they want to be done by provider agents. These jobs may require more types of resources than a single agent can provide, hence providers need to collaborate. All agents are self-interested entities that do not necessarily have a common goal; we allow them to be designed and owned by different parties, which limits the possibilities for global control in this setting. In addition, we assume that provider agents do not know anything about the future orders of the customers.

If the system contains a large set of agents or the jobs to be assigned require the collaboration of many providers, assigning the jobs with standard auction systems is very time consuming and methods to improve the performance of the system become more important. If the same type of job needs to be assigned many times or parts of the jobs are the same over and over again, the system is be more efficient if this repeating structure on the demand side is reflected by an anticipating structure on the side of the provider's. That is, providers who are successful at completing a job should form relationships that facilitate long-term teamwork. This provider grouping can be formalised by the concept of *organisation*. Organisations are social structures that provide processes for conflict resolution, which results from previously resolved problems or conflicts [7]. They institutionalise anticipated coordination, which is especially useful for medium- and large-scale applications that require the limitation of the agents' communication behaviour.

Jennings writes that "the development of robust and scalable software systems requires autonomous agents that can complete their objectives while situated in a dynamic and uncertain environment, that can engage in rich, high-level social interactions,

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 81–90, 2003.

and that can operate within flexible organisational structures" [11]. Agents acting in an organisational structure can encapsulate complexity of the subsystems (simplifying representation and design) and modularise functionality (providing the basis for rapid development and incremental deployment).

To model these aspects, we use the concept of a *holonic agent* or *holon* as introduced in [6,8]. The concept is inspired by the idea of recursive or self-similar structures in biological systems [13]. A *superholon* consists of parts called *subholons*, which in turn may be superholons themselves, thus introducing recursion as a modelling technique. Any holon that is part of a whole is thought to contribute to achieving the goals of this superior whole. To the outside, each holon is represented by a distinguished *head* which co-ordinates the activities of the holon. Apart from the head, each holon consist of a (possibly empty) set of other agents, called *body agents*. The holonic agent may have capabilities that emerge from the composition of its agents and it may have actions at its disposal that none of its agents could perform alone. Body agents can give up part of their autonomy to the holon to enhance it's overall performance. Committing to the participation in a holon, agents limit their possible future actions and therefore give up part of their autonomy. The degree to which they give up their autonomy is not fixed in advance but depends on the circumstances and is subject to negotiation between the agents participating in a holon. The least sacrifice of autonomy can be seen in holons forming a loose *federation* of agents. The long-term commitment in this form is actually so low that agents need to negotiate their coordination on a case-by-case basis. The federation is at one end of the autonomy spectrum and does not differ significantly from conventional multiagent systems (it merely institutionalises preference structures). At the other end of the spectrum, agents can give up all of their autonomy and merge into a single agent. Between the two extremes, there can be hybrid forms of different nuances.

However, up to this point little work has been done to elaborate these nuances. The contribution of this paper is to define a number of such hybrid forms motivated by inter-organisational networks found in human societies (Section 2) and investigate their properties with respect to the autonomy of involved agents (Section 3).

2 Organisational Networks

2.1 The Matrix of Delegation – A Grammar for MAS Organisation

Recent work on delegation (see e.g. [4] for an extensive treatment) shows that this is an interesting concept highly relevant for multiagent systems. The mechanism of delegation makes it possible to pass on tasks (e.g. creating a plan for a certain goal, extracting information) to other individuals and furthermore, it allows for the specialisation of these individuals for certain tasks (functional differentiation and role performance, etc.). Representing groups or teams is also an essential mechanism for social processes of organisation, coordination and structuring. We distinguish two types of delegation: task delegation and social delegation. We call the procedure of appointing an agent as representative for a group of agents *social delegation*.

Social delegation is in several respects different from the well-known task delegation. For example it involves a possibly long-term dependency between delegate and

represented agent, and the fact that another agent speaks for the represented agent may incur commitments in the future, that are not under the control of the represented agent. Social delegation is more concerned with the delegate performing a certain role, than with producing a specified product. In holonic terms, representation is the job of the head, which can also be distributed according to a set of tasks to different agents. Just like fat trees (multiple bypasses to critical communication channels) in massive parallel computing, distributing the task of communicating to the outside is able to resolve bottlenecks. This makes social delegation a principle action in the context of flexible holons and provides the basic functionality for self-organisation and decentralised control.

Thus, we believe it is justified to differentiate two types of delegation: task delegation, which is the delegation of (autistic, non-social) goals to be achieved and social delegation, which does not create a solution or a product but represents a set of agents. Both types of delegation are essential for organisations, as they support independence from particular individuals through task and social delegation.

Table 1. The delegation matrix showing two modes of delegation and four mechanisms for performing each mode. Theoretically, every combination of mode and mechanism is possible in multiagent organisation.

	Task Delegation	Social Delegation
Economic Exchange		
Gift Exchange		
Authority		
Voting		

Given the two types of delegation, it remains to be shown been how the action of delegation is performed. We observe four distinct mechanisms for delegation (see also Table 1):
(i) Economic exchange is the standard mode in markets: the delegate is paid for the delegated task or representation. In economic exchange, a task is exchanged for money, where the parties involved assume that the value of both is of appropriate similarity.
(ii) Gift exchange, as an important sociological mechanism [1], is a deliberate deviation from the economic exchange. The motivation for the gift exchange is the expectation of either reciprocation or the refusal of reciprocation. Both are indications about the state of the relationship between the involved parties. This kind of exchange entails risk, trust, and the possibility of conflicts (continually no reciprocation) and the need for an explicit management of relationships within an agent. The aim of this mechanism is to accumulate strength in a relationship that may pay off in the future.
(iii) Authority is a well known mechanism, it represents the method of organisation used in distributed problem solving. It implies a non-cyclic set of power relationships between agents, along which delegation is performed. However, in our framework authority relationships are not determined at design time, but they are the result of an agent deciding at runtime to give up autonomy and to allow another agent to exert power.
(iv) Another well-known mechanism is voting, whereby a number of equals determine that one of them is the delegate by some voting mechanism (majority, two thirds, etc.).

Description of the mandate (permissions and obligations) and the particular circumstances of the voting mechanism (registering of candidates, quorum) are integral parts of the operational description of this mechanism and must be accessible to all participants.

As suggested in Table 1, all four mechanisms work for both types of delegation: for example, economic exchange can be used for social delegation as well as for task delegation. This set of mechanisms is not necessarily complete, however, many mechanisms observed in human organisations that seem not to be covered here, are a combination of the above mechanisms.

2.2 The Spectrum of Organisational Forms

In general, we allow agents to be members of several organisations at the same time. In order to unambiguously determine which organisation is responsible for an incoming order, this general rule is restricted to all organisations an agent is engaged in being created for different types of orders. We will now describe (building on the matrix of delegation from the previous section) seven different forms of organisation and non-organisation for MAS in the order of increasing coupling between agents along a spectrum. Organisations may differ as agents interact either in a cooperative, in a competitive or in a authoritarian way (c.f. for example [14]) The names for the different forms are derived from the types of firms that are typically investigated in the field of organisational sociology.

Single, Autonomous Agents: This form of coordination is not particularly relevant but it provides the theoretical starting point, with fully uncoupled agents. All agents that provide services do not interact with each other to accomplish their tasks, the only interaction is between providers and customers.

Market: In the market-style interaction, agents exchange jobs directly and there is some kind of payoff (here represented as money). This does not necessarily imply that agents build a relationship or an organisation in the strict sense, as interaction is short term, case by case based. The provider agent that re-delegates parts of a job acts as the holon head for this specific job.

Virtual Enterprise: The virtual enterprise is a temporary network of legally independent companies to share skills, costs and access to each other's market. Virtual enterprises promise to offer the best of both worlds, flexibility and economy of scale. They are networks of legally and economically independent enterprises, each concentrating on its core competencies and out-sourcing the rest, modelled on the best-of-breed organisation. The virtual enterprise appears and acts like a single enterprise to the outside has to do world [2]. Moreover, there is no physical institutionalisation of central management functions. The contract defining the relationship between the participating enterprises is deliberately left loose, in order to facilitate quick formation and greater flexibility in re-organisation. In our model, a virtual enterprise consists of provider agents with equal rights, there is no single designated head agent. A virtual enterprise is product-specific. Each member agent may accept jobs, but must start a new internal auction for each of its subtypes among its partners. This member agent becomes the head of the virtual enterprise for this specific job, other members may be heads of the holon for other jobs.

There is no specific profit distribution other than the normal negotiation in the course of the internal auctions.

Alliance: An alliance as an organisational type is different than the virtual enterprise because of a long term contract between the participants that regulates a closer cooperation [9]. The relationship between the companies is formalised by a contract, which is the result of negotiation between the different companies. Alliances are not fully integrated economically and legally, therefore the profit distribution for all internal transactions is regulated in advance. Alliances are founded in order to create at least one new product. As the companies are only partially integrated they usually supply other products apart from the alliance as well. Thus, they are generally allowed to join other organisations apart from those which produce the same product as the alliance. As alliances are in some way legally integrated they need to appoint at least one CEO (representative), which is done by voting. The representation of the alliance incurs valuable reputation and contact to customer agents, hence it implies (economic) power. Quitting of one of the agents with many customer contacts may cause loss to the organisation, as customers may prefer to interact with the provider agent they already are acquainted with, no matter in which organisation it is in. To decrease the incentive to join the alliance solely for this purpose and for the stability of the organisation, and a focal participant, who is, due to his already powerful position, not reliant on this increase in reputation, is appointed by *social delegation* through *voting* to represent the alliance. The profit is distributed among the head (representative) and all body agents necessary for performing the task by using *economic exchange* and *gift exchange*. However on creation of the alliance agents agree on a ratio (which is in our case fixed by the designer) that describes how profit is split between the head agent and the body agents that are involved in performing the task.

Strategic Network: Strategic networks differ from virtual enterprises in that they use stronger legal contracts, and feature a *hub firm* that sets up the network, and takes pro-actively care of it [10]. The hub firm in a strategic network is usually significantly larger than the other members of the network. It coordinates activities in the strategic network, but the members retain their legal independence and autonomy. This network arrangement allows a participating firm to specialise in those activities of the value chain that are essential to its competitive advantage, reaping all the benefits of specialisation, focus, and, possibly, size. The time frame and financial volume are usually larger than in the case of virtual enterprises, but firms have still the right to leave the network.

In our model, strategic networks consist of a head agent and body agents. If an incoming order matches the product of the strategic network, the rules that apply to the receiving agent discriminate whether it is the head or a body agent. Body agents may not directly accept a *call for proposals* (`cfp`) from outside, but must *bounce* it. Bouncing means that they refuse the order, but they send the name of their head inside the refusal message so the sender can resend the `cfp` to the head instead. The semantics of bouncing is that the head of the organisation is the one responsible for the organisation's interaction to the outside and all inter-organisational communication must be channelled through him. Heads can accept orders from the outside. They know about their body agents' schedules and resources, and can instruct them to do a job at any given time. Strategic networks are product-specific, so multiple memberships are allowed. The profit distribution is according to a fixed ratio.

Group: Groups are formed from enterprises that retain their legal independence, but are bound by contract to the authority of the central firm. Here, we mean group as the organisational structure of a firm as in "Bertelsmann group", not in the socio-psychological meaning of "team". In contrast to the strategic network, no multiple memberships are allowed, and usually there is no exit option for subordinate firms. All economic activities are focused on the group and subject to directions from the head enterprise. The interdependency between the firms is found in an authoritative hierarchy.

In our model, an agent who is a member of a group is not allowed to be a member of any other organisation. Body agents have to bounce incoming orders. Head agents may order body agents to do a specific job. This inclusion of all economic activity in the group results in the head agent always being up-to-date about its body agents' resource allocations. The head agent retains all the profit for orders completed by the group. Every round, it pays a fixed amount of money to each body agent.

Corporation: A corporation is the result of the complete inclusion of all legal and economic aspects of the original companies into a new entity. This organisational form marks the other extreme of the spectrum between market and hierarchy. Companies merging into a corporation give up all their autonomy. The process is usually not reversible; once inside a corporation, the former status cannot be regained. In the business world, the process of merging usually happens when a large company assimilates a much smaller one. We model corporations by letting the head assimilate the resources of its body agents. After the assimilation, the body agents are removed from the simulation. The head then acts like a normal single agent, except that it does not form new organisations.

We presented here multiagent organisations starting with the most autonomous form and proceeded to the one with least autonomy. The model provides a framework for the agents' decision at runtime. In theory, each agent can choose, depending on the situation in the MAS, whether it is in its interest to change its current status. As each organisational type has advantages and disadvantages, it may well be that a transition is not beneficial in the light of the current market situation.

2.3 Synopsis

Before we discuss the effect of the model on agent autonomy, we will explain the summarising table of the critical features of the organisational forms. Table 2 gives a synopsis on the organisational types, it characterises organisational forms in terms of their properties: the mechanism for task delegation (TD), social delegation (SD), membership limitations (M), profit distribution (PD), and the role of the holon head (HH).

Membership limitations can have the value "limitation on product", which means that the agent is free to choose other organisations to join, as long as they do not use the same set of resources. This parameter can also denote that there is no limitation (as in the market) or that an agent is only allowed to be a member of one single organisation (as with the group).

Profit distribution is on a per job basis using economic exchange, a fixed ratio between head and body agents (e.g. 20:80), or a fixed income which is paid by the head to the body agents regardless of the number of jobs performed (in this case, variable costs are

Table 2. Overview of the five types of holonic organisation. Rows specify for each type the dependence introduced (Dep), membership limitations (M), the mode of profit distribution (PD), the role of the holon head (HH), and the protocol used (P).

	Market	Virtual Enterprise	Alliance	Strategic Network	Group	Corporation
M	No Limitation	Limitation on Product	Limitation on Product	Exclusive Membership	Exclusive Membership	N/A
PD	Case by case	Case by case	Regulation	Fixed Income	N/A	N/A
HH	One/All	All	One	One	One	One
TD	Economic Exchange	Economic/ Gift Exchange	Economic/ Gift Exchange	Authority	Authority	N/A
SD	Economic Exchange	Economic/ Gift Exchange	Voting	Authority	Authority	Authority
P	HCNCP	HCNCP	DCP	DCP	DP	None

paid by the head plus a fixed income chosen by the designer). The details of the fixed ratio and fixed income must be agreed on at the time of creating the organisation. There is no profit distribution in the corporation, because the original agents have merged into a single entity.

The role or number of the holon heads (HH) for the market is to some extent up to interpretation: Although there is only one job per holon and only one agent communicating and coordinating for this holon, all agents in the system are allowed to accept jobs and then engage in coordination and communication. In the virtual enterprise all agents can receive incoming jobs and redistribute them. All other forms allow only a single point of access to the outside world. As shown in the table, depending on the organisational type we use three different protocols. The mechanisms for task delegation (TD) and for social delegation (SD) have been described in Section 2.1. We will come back later to the protocol used for task-assignment internal to the organisation (P).

3 Implications of Organisational Structure on Agent Autonomy

According to Castelfranchi [3], autonomy is not only a quantitative dimension (a question of more or less), but divided into several dimensions and it directly corresponds to (the lack of) dependence in some aspect. Some of these dimensions map precisely to the differences between organisations in our discussion. Along the spectrum we will now identify the dimensions of autonomy that are given up in the different organisational forms. Our theoretical starting point are the single, non-cooperating agents, which are fully autonomous.

Skill and resource autonomy: The agent that enters a pure market relationship divides a job and re-delegates those parts it cannot perform by itself. Therefore, it is dependent on the *skill* and *resources* of other agents, and thus loses skill and resource autonomy (emphasised words correspond to Castelfranchi's terminology).

Conditional autonomy: When creating the virtual enterprise the agents specify which compound product this organisation is designed for. Entering the organisation implies that the agent commits to provide its resources for this compound product and that any

incoming orders will only be shared with other agents in the organisation. Hence, it may only collaborate with other agents under the condition that the result of this collaboration is not of the same type as the product of the organisation. Agents cannot arbitrarily choose collaborators, and therefore give up *conditional autonomy*.

Representational autonomy: The creation of an alliance requires the election of a single representative, which will fully take over communication to the outside of the organisation. No other agent is allowed to accept orders from other agents, they may not represent themselves, and hence lose *representational autonomy*.

Goal dynamics autonomy: Agents in a strategic network must announce the cost function and are subordinates to a single representative, which has the power to order them to perform tasks. This means that they cannot fully influence their own set of goals, and they lose *goal dynamics autonomy*.

Planning autonomy: The representative of a group has complete knowledge of the schedule of its subordinate agents. Therefore, it is not necessary to communicate for determining an agent that can perform a task. Planning can be done centrally by the representative, which then merely informs subordinates about when to perform which task. These agents have lost *planning autonomy*.

Processing autonomy: When agents have merged into a corporation agent, they give up individual computational resources and lose the last remaining autonomy, namely *processing autonomy*.

It is worth noting that each form of organisation builds on the previously described form, and introduces new dependencies corresponding to a loss of autonomy of the body agent. Therefore, we can speak of a total ordering of the organisational type and hence, a *spectrum* of organisational types.

This discussion is not just academic, but it has practical impact: As the basis for task-assignment we apply the *Holonic Contract Net with Confirmation Protocol (HCNCP)*. The HCNCP (see Figure 1) uses the standards of the *Foundation for Intelligent Physical Agents (FIPA)* as a reference (cf. [5]). It extends Smith's contract-net protocol (CNET) [15] and avoids the problem of committing too early, which often leads to sub-optimal outcomes in the CNET, and can be applied in a cascading manner (for more detailed discussions see [12]). In the best case, the HCNCP requires six messages between a contractor and a contractee to assign a job. The HCNCP is used for inter-organisational communication, market interaction, and intra-organisational communication in the organisation form *virtual enterprise*. However, if agents are inside an *alliance* or a *strategic network* the initiator of an intra-organisational task-assignment protocol can save sending a cfp as a consequence of the loss of representational autonomy It can directly proceed to the request for confirmation of being able to perform the task (Direction with confirmation protocol, DCP, cf. Figure 2). Although there is an authority relationship between representative and all other agents, they may still be a member of several organisations, and hence, the schedule of an agent is not fully known by the representative. In the group however, this is the case, and therefore a further part of the task assignment protocol can be saved (Direction protocol, DP, cf. Figure 3). Obviously, the different protocols have implications on the overall communication effort and are a direct result of the different levels of autonomy. In order to evaluate our organisational structures, we started an empirical study of the communication patterns

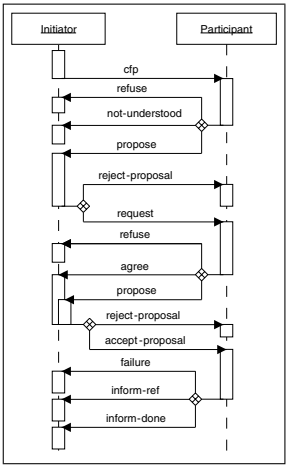


Fig. 1. The Holonic Contract-Net with Confirmation Protocol (HCNCP) which is used as the default protocol for efficient and cascading task assignment.

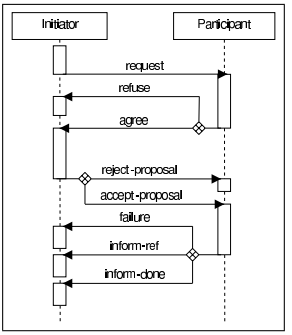


Fig. 2. The Direction with Confirmation Protocol (DCP) which is used in the strategic network, saves the announcement of cost proposals.

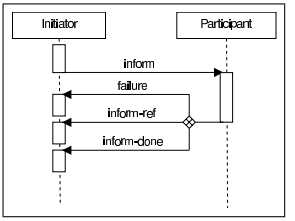


Fig. 3. The Direction Protocol (DP) which is used in the group, makes use of the knowledge of the initiator and the authority relationship.

inside of each of these organisations given a fixed task. In a round-based simulation, customers were created to announce a task that can consisted of three sub-tasks of types `A`, `B`, and `C`. For each organisation we tested one scenario, which contains 20 provider agents for each of the three types, each type occurring with production costs 4, 5, and 6 (a total of 180 provider agents). There are 60 customer agents emitting one order of type `ABC` each round. In each configuration, all occurring organisations are of the same form. Each organisation has three member agents, one for each of the three task types. One member agent has cost 4, one cost 5, and one cost 6. Each experiment ran for one hundred rounds. In this setting the aforementioned differences result in a diversification of the amount of communication required by each organisational form for solving the same task-assignment problem. For this setting, the number of messages for the virtual enterprise (ca. 14000) is higher than that for the strategic network (ca. 4200), which in turn is higher than that for the group (3.200). Corporation scenarios have the lowest number of messages (ca. 2200). The number of messages for agents in a pure market relationship is one order of magnitude higher (ca. 256000).

4 Conclusion

Multiagent systems have the power to model forms of collaboration inspired by real world organisations in a natural fashion and we contributed by laying out a framework of different organisational forms relative to the following parameters: mechanisms for task and social delegation, membership, profit distribution, number of representatives and protocol for task-assignment. This theory of how agents of different capabilities can be

tied together is inspired by well-known sociological descriptions of inter-organisational networks and involves a distinct description of dimensions of autonomy and dependence agents have in these networks. While increasing coupling between agents increases performance in terms of numbers of messages, this coupling comes at a price. It requires the loss of skill/resource autonomy, goal dynamics autonomy, representational, planning, but be and finally, processing autonomy. So, the choice of an organisational form for a concrete application not only depends on the performance requirements but also on the necessity of privacy and the extent to which local decision-making is a requirement, a decision that involves considering a complex trade-off.

Acknowledgements. This work was funded by the Deutsche Forschungsgemeinschaft in the priority program *Sozionik* under contract Si 372/9-2. We are indebted to sociologists Dr. Michael Florian, Dr. Frank Hillebrandt, Bettina Fley and Daniela Spresny for many inspiring discussions. We also thank the anonymous reviewers of the conference for comments and advice.

References

1. P. Bourdieu. *Pascalian Meditations*. Polity Press, Cambridge, Eng., Oxford, Eng., 2000.
2. L.M. Camarinha-Matos, H. Afsarmanesh, C. Garita, and C. Lima. Towards an architecture for virtual enterprises. *Journal of Intelligent Manufacturing*, 9(2):189–199, 1998.
3. C. Castelfranchi. Founding agent's "autonomy" on dependence theory. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, pages 353–357, 2000.
4. C. Castelfranchi and R. Falcone. Towards a theory of delegation for agent-based systems. *Robotics and Autonomous Systems*, 24:141–157, 1998.
5. FIPA. Foundation for Intelligent Agents http://www.fipa.org/repository/ips.html, 2002.
6. K. Fischer. Holonic multiagent systems — theory and applications. In *Proceedings of the 9th Portuguese Conference on Progress in Artificial Intelligence (EPIA-99), LNAI Volume 1695*, pages 34–48. Springer-Verlag, 1999.
7. L. Gasser. Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence*, 47:107–138, 1991.
8. C. Gerber, J. Siekmann, and G. Vierke. Flexible autonomy in holonic multiagent systems. In *AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.
9. R. Gulati and Martin Garguilo. Where do interorganizational networks come from? *American Journal of Sociology*, 104:1439–1493, 1999.
10. J. Carlos Jarillo. On strategic networks. *Strategic Management Journal*, 9:31–41, 1988.
11. N.R. Jennings. Agent-based computing: Promise and perils. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1429–1436, 1999.
12. T. Knabe, M. Schillo, and K. Fischer. Improvements to the FIPA contract net protocol for performance increase and cascading applications. In *International Workshop for Multi-Agent Interoperability at the German Conference on AI (KI-2002)*, 2002.
13. A. Koestler. *The Ghost in the Machine*. Hutchinson & Co, London, 1967.
14. Walter W. Powell. Neither market nor hierarchy. network forms of organization. *Research in Organizational Behavior*, 12:295–336, 1990.
15. R. G. Smith. The contract net: A formalism for the control of distributed problem solving. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, page 472, 1977.

Fault-Tolerant Behaviour in Holonic Manufacturing Systems: A Simulation Study

Thomas Neligwa and S. Misbah Deen

Department of Computer Science, Keele University
Staffordshire, ST5 5BG, United Kingdom
{neligwa, deen} @cs.keele.ac.uk

Abstract. This paper extends our earlier work on the fault-tolerant HMS architecture and its operational model in which we also proposed an operational strategy called the temperature model. This framework allows holons to exercise their full autonomy, individualism and adaptability as their primary behaviour but it implicitly enables the HMS to exhibit its characteristic features, viz. flexibility and robustness as an emergent behaviour resulting from the primary behaviour. The paper presents a simulation study of this HMS environment in order to demonstrate empirically, the benefits of holonic cooperation. Our simulation model is characterised with an unprecedented number and mix of tasks, and random faults on manufacturing holons as is expected for a realistic, error-prone manufacturing environment. Experimental results generated from this simulation have shown how some performance metrics of the HMS converge to within the expected tolerance.

1 Introduction

A holonic manufacturing approach distinguishes itself from the rest of manufacturing approaches by offering considerably higher degrees of flexibility and robustness. Flexibility in the context of a Holonic Manufacturing System (HMS) refers to its ability to accommodate low-volume, high-variety production requirements. Robustness ensures that the manufacturing system reacts and adapts dynamically to adverse events during operation. Such events include transportation delays, lost communication messages, mechanical faults on production equipment and unprecedented new requirements, to mention a few. The HMS environment is therefore expected to tolerate disruptions, and therefore maintaining the desired level of throughput and reducing production overheads and waste. In the HMS context, flexibility and robustness can be regarded not only as characteristic operational features but also performance benchmarks of Holonic Manufacturing Systems.

In order to achieve these operational objectives, Holonic Manufacturing Systems are modelled such that a number of holons cooperate using an appropriate strategy to solve and control the distributed manufacturing tasks. Each holon is an autonomous entity with social characteristics such as communication, cooperation, etc. Because individual holons can possibly have their own individualistic goals, such as maximising their utility or profit, cooperation strategies are carefully designed so that the

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 91–99, 2003.

goals of the cooperating system as a whole can be reasonably satisfied while holons are pursuing their individual goals. This paper extends our earlier work on the fault-tolerant HMS architecture and its operational model in which we also proposed an operational strategy called the temperature model [1]. This framework allows holons to exercise their full autonomy, individualism and adaptability as their primary behaviour but it implicitly enables the HMS to exhibit its characteristic features, viz. flexibility and robustness as an emergent behaviour resulting from the primary behaviour. As a matter of fact, this HMS environment is a highly distributed system with unprecedented non-linear interactions between holons.

Studying the operational behaviour of a highly distributed system characterised with non-linear interactions is not trivial because interactions can be computationally intractable. In general, when large numbers of cooperating agents (or holons in HMS) use adaptive rather than optimising strategies, predicting the consequences of dynamic interactions becomes too difficult for mathematical tools to yield satisfactory results in limited time [2]. In other words, we cannot effectively study an ideal HMS by using mathematical formalisms. The usual practice (in studying autonomous multi-agent systems) is to specify some simple rules on how agents should behave and interact, and then observe in a simulation, the emergent properties that occur at the system level. We have therefore adopted this approach to studying the fault-tolerant HMS environment. Our simulation model is characterised with an unprecedented number and mix of tasks and random faults on manufacturing holons as is expected for a realistic, error-prone manufacturing environment. Experimental results generated from this simulation have shown how the measures of effectiveness and robustness converge to within the expected tolerance.

The rest of the paper is organised as follows: Section 2 presents an overview of the fault-tolerant HMS environment which forms the basis of our simulation model. Section 3 discusses the simulation study and presents a summary of the experimental results. Section 4 concludes the paper.

2 Fault-Tolerant HMS Environment

Because the fault-tolerant HMS model used as a basis for this simulation study has been extensively covered in our earlier papers: [1], [3], [4] and [5], we shall only present here its brief overview, for the completeness of this paper. We will emphasize on three main elements: the application model, co-operation model and the operational model. Note that we have deliberately disregarded discussing the intrinsic design and implementation of individual holons as we are more interested on the operational behaviour of the cooperating system, in this paper.

2.1 Application Model

The HMS philosophy requires distribution, autonomy and co-operation of manufacturing resources in order to achieve the desired levels of flexibility, adaptability and robustness. This requires modelling the manufacturing process hierarchy (with their

respective resources, i.e. holons, as a distributed system of co-operating holons. Each major production entity in the shopfloor, e.g. an individual machine or a group of machines, warehouses, turntables, conveyor belts, automated guided vehicles, humans, etc., can be modelled as skill holons. Skill holons of the same (or redundant) skill capability are often grouped together into a skill class and are referred to as twins, with each class having one minder holon which is responsible for the welfare of the twins. There are several advantages of introducing twins into the production environment. First, twins can be used to process jobs in parallel in order to enhance throughput. Second, twins can be used to undertake jobs orphaned by broken down twins in order to minimise the impact of malfunctions. And third, twins can equally share the workload among themselves so that none of them can become overloaded. These are important features for a fault-tolerant HMS.

In order to support these features, an ideal HMS shopfloor would be organised such that twins can transfer jobs between each other via a suitable transportation link, e.g. a conveyor belt or an automated guided vehicle. It is also possible for a skill holon to have more than one skill, and hence belong to more than one skill class. Having multiple skills is another important HMS feature required for enhancing flexibility, which is highly desired for the production of low-volume, high variety merchandise, with minimum reconfiguration overheads.

2.2 Cooperation Model

Co-operation is a process through which holons interacting via a *Co-operation Domain* (CD) execute dependent activities of a joint task. The CD is a physical infrastructure through which a family of skill holons interacts and shares information. Because the CD does not explicitly support the dynamic interactions of cooperating holons and the relevant task execution information, we prefer to use a higher-level infrastructure known as a *Co-operation Block* (CB).

The CB is a logical structure denoting a dynamic alliance between holons interacting through a CD. Every CB has a *co-ordinator* holon and at least one member holon or *cohort*. The co-ordinator of each CB is responsible for receiving tasks from external users or other holons and decomposing them into appropriate subtasks, initiating negotiations for scheduling or rescheduling, and monitoring the execution by handling the dynamic task constraints and disruptive events (in co-operation with cohorts). Several CBs may be formed dynamically, each for one joint task and have a finite life equal to the duration of the task for which they were formed, i.e. a CB is a dynamic entity, created for the execution of that task and destroyed when the task is completed.

2.3 Operational Model

The operational model is mainly concerned with the dynamic interactions between holons, communication strategies and the processing of tasks (scheduling, execution and rescheduling). These concepts are briefly discussed below.

Task scheduling

In the ideal HMS world, scheduling is done in real time, and hence requires real-time data from co-operating holons. This can be achieved by using the Contract Net proto-

col [6], which is the most popular interaction strategy for co-operative systems. For each task, the relevant co-ordinator decomposes it into relevant subtasks and initiates negotiations with cohorts for scheduling, i.e. assign each activity to specific skill holon along with timing constraints. The coordinator can start by inviting bids from potential cohorts for the entire set of activities and then construct the preliminary schedule based on the proposed start times and precedence constraints. The schedule consisting of earliest-start (ES) times and a latest start (LS) times, can be calculated by using the following well-known algorithm [7]:

$$ES_{j(k)} = Max\left\{ES_{i(k)} + d(k)\right\} \quad for\ all\ i \quad \text{and}$$

$$LS_{j(k)} = Min\left\{LS_{i(k)} - d(k)\right\} \quad for\ all\ j$$

where $d(k)$ = the duration of activity k, $i(k)$ = an array of previous activities, and $j(k)$ = an array of next activities. This algorithm generates a matrix for each activity with $\{[ES, LS], [EF, LS]\}$ where

$$LF = LS + d(k) \quad \text{and} \quad EF = ES + d(k)$$

After finalising the preliminary schedule, the coordinator must ask all successful bidders (cohorts) to confirm their prescheduled times. If the prescheduled time is unacceptable, a newer or approximate bid must be submitted from that class via the minder, iteratively until an acceptable time is found. In effect each iteration affects all succeeding activities (forward-chaining), which must be allocated new times as appropriate. In the event that no suitable time is found for a particular activity, the coordinator might abort the scheduling session to start afresh or invoke an alternative decomposition. This approach to scheduling dependent activities cooperatively was developed during the course of this study, and we have called it an *incremental scheduling strategy*.

Ideally, each skill holon may decompose their activities into a lower levels as suitable for processing but they do so with their full autonomy and discretion without involving the coordinator.

Operations Scheduling

The fact that each twin bids for its own activities obliges them to maintain their own local schedules and execution statuses. These can simply be interpreted as a sequence of operations required for completing various activities, each from possibly a different task. Should it be necessary to reschedule these operations locally in order to satisfy internal or external requirements, this can be done discreetly so as to minimise disruptions at a higher level. In fact, since each activity/operation has an *ES* and *LS* as lower and upper time limits, respectively, a new time, t can be assigned to a particular operation, such that $ES \leq t \leq LS$. The difference between LS and ES is known as *slack* time in Operations Research literature.

Execution
Whenever the task schedule becomes due, the coordinator, minder and the responsible twin will initiate execution. Execution of task schedules is conditional so that precedence constraints can be effectively enforced. An activity cannot begin execution even when the schedule is due unless its precondition is true, which signifies that a predecessor activity has been completed. For activities without precedent constraints, preconditions are determined externally. On completion of each activity, the twin executes its end condition by sending a precondition to all its dependent activities.

Sending and receiving preconditions is crucial for the conditional execution of tasks because if any of messages is lost, the execution chain breaks down and all activities on the chain would be blocked. It is therefore imperative that a very robust messaging infrastructure is used for real-time communications and message exchange.

Temperature Model and Rescheduling
In the study of thermodynamics, it is well known that if two objects are placed in thermal contact with each other, the temperature of the warmer body decreases while the temperature of the cooler body increases, until a common temperature is reached somewhere between the two initial temperatures. At this temperature, the two objects are said to be in thermal equilibrium, i.e. the two objects cease to have any energy exchange due to differences in their temperature.

We have used these concepts as a metaphor to describe a rescheduling strategy for HMS applications. Temperature is used to gauge the load levels of individual holons as follows: overloaded holons are regarded as ‘hot’ (or ‘infinitely hot’ if broken), and idle or underloaded holons are regarded as ‘cold’ or 'cool' respectively. According to the principles of thermodynamics, hot twins will seek to dissipate their heat towards cooler twins and vice versa, thus sharing their workload and leading the system into equilibrium, i.e., uniform workload distribution. In the HMS environment temperature is induced by run-time delays due to breakdowns, overloads, etc. A formal description of the model using three basic concepts of thermodynamics, viz., *relative heat, temperature and latent heat* can be found in [1].

3 Simulation Study

The overall aim of the simulation is to show quantitatively the operational behaviour of the envisaged HMS environment as outlined earlier. Firstly, we have estimated the system robustness – by rescheduling activities orphaned by broken down twins and those affected by delays due to precedence constraints, and therefore minimising the impact of malfunctions, i.e. the HMS tolerates disruptions and recovers gracefully from malfunctions. In effect, some tasks which would otherwise have faced lengthy delays can be completed earlier than expected (see Section 3.1). Secondly, we have shown how rescheduling can improve the utilisation of individual holons, by using time slots which could otherwise be wasted (see Section 3.2). A useful by-product of our simulation study is the relationship between transfer costs and waiting costs,

which can be used to predict the operational costs of the system for a given period of time. This measure is discussed in Section 3.3.

We have presented below a summary of the experimental results based on a hypothetical testbed with seven skill classes, each with five twins. Each twin calculates its variable parameters, for example, the number of slots, anticipated breakdown sequence, etc., with respect to the given simulation length. Twin slot size ranges from 120 seconds to 360 seconds. We have also simulated a fairly complex mix of task decompositions with random arrivals at their respective coordinators for scheduling. The results below are compared over ten simulation runs, each with a length of 28400 seconds (7.9 hours), hence the large figures. The latter were necessary for drawing conclusive results. Note that all numeric values have been arbitrarily chosen to illustrate a typical manufacturing case.

3.1 Improvement on Completion Times

There is an interesting comparison of prescheduled completion against the actual completion times. Here prescheduled completion includes all the actual delays and the actual completion refers to the completion after applying the temperature model. We have exaggerated the random arrivals and failures so that the significance of the model can be clearly pronounced. Fig. 1(a) shows an exaggerated delay of nearly 6000% on prescheduled completion time but the fault tolerant model improves the completion time by, $\gamma \approx 57\%$ as shown in Fig. 1(b).

3.2 Utility Gain with Temperature Model through Recovery and Cooling

We compared the number of slots that received jobs under normal operation to the number of slots that received jobs as a result of rescheduling in order to establish whether the temperature model brings about a significant improvement. Over ten simulation runs, the temperature model improves the utilisation of skill holons by a constant value for a fixed set of parameters (system configuration).

Fig. 2 shows the total number of slots that would have been used without rescheduling in tandem with the number of slot that were used after rescheduling with the temperature model for ten simulation runs. These results show that the ratio of the two variables is a constant. If i and I represent the number of idle slots with and without rescheduling respectively, we can estimate the ratio between the two as: $\beta = i/I$. Based on the random arrivals and random malfunctions used for the simulation runs, from Fig. 2, $\beta = \mathbf{34\%}$**,** which is a huge improvement in the utilisation of the system. However, the magnitude of this factor is entirely dependent on the arrival and failure rates.

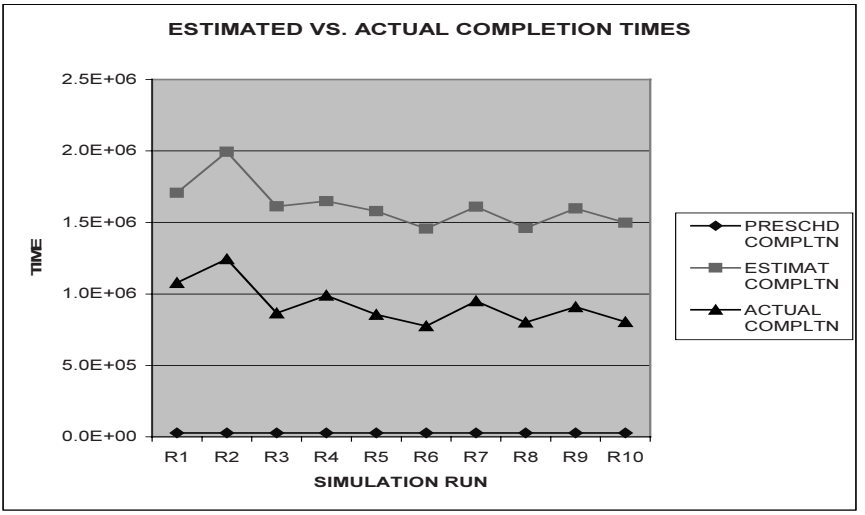


Fig. 1(a). Estimated vs. actual completion times

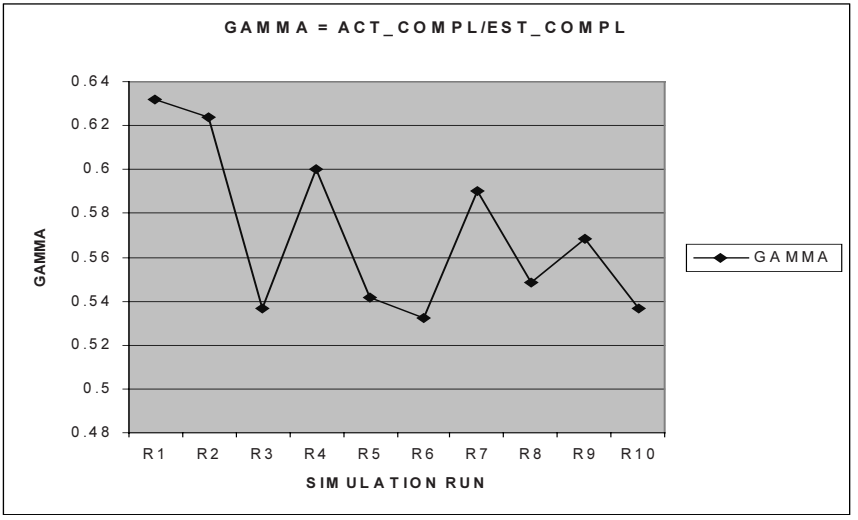


Fig. 1(b). Gamma $\approx$ 57% over 10 simulation runs.

3.3 A Comparison of Waiting and Transfer Costs

It is also possible to compare the waiting costs vs. transfer costs per class or the entire system, per operational session in order to establish relationship between these two

quantities. The ratio, say, δ = *transfer costs*/*waiting costs*, can be used to extrapolate both waiting costs and transfer costs if all other parameters in the system are kept constant or in a fixed range. Alternatively, δ may be used to determine the amount time a particular task can wait without incurring undue cost. The simulation results comparing transfer costs vs. waiting costs over ten runs are shown in Fig. 3. Note that waiting costs are relatively higher than the transfer costs as discussed earlier but the ratio between the two quantities is constant for a given set of simulation parameters. In this particular case (Fig. 3), • is constant (≈ 29%) over 10 simulation runs.

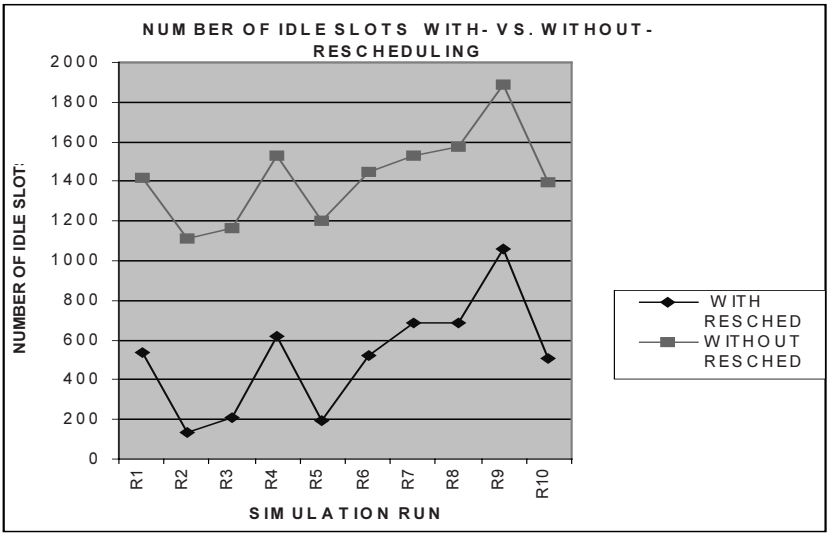


Fig. 2. Number of slots used without rescheduling vs. with rescheduling

4 Conclusion

It is apparent that the fault-tolerant HMS model introduces significant computational overheads but these are negligible in comparison to the physical operations in the HMS environment. In addition, because waiting costs in production are relatively higher than transfer costs, the latter can always be offset by the benefits of rescheduling. The model shows significant robustness and tolerance to random faults without compromising the ability to dynamically schedule multiple tasks and improve the throughput continually. We have used purely random numbers to simulate a hypothetical manufacturing testbed but the results are strikingly conclusive. Because the model is computationally intensive and highly dependent on messages, it is imperative that a very reliable messaging mechanism is used. We used Java for these experiments but we experienced lots of lost messages. Many activities incurred unneces-

sary delays due to missing information. Certainly, this had some effect on our final statistics but the patterns are clearly evident.

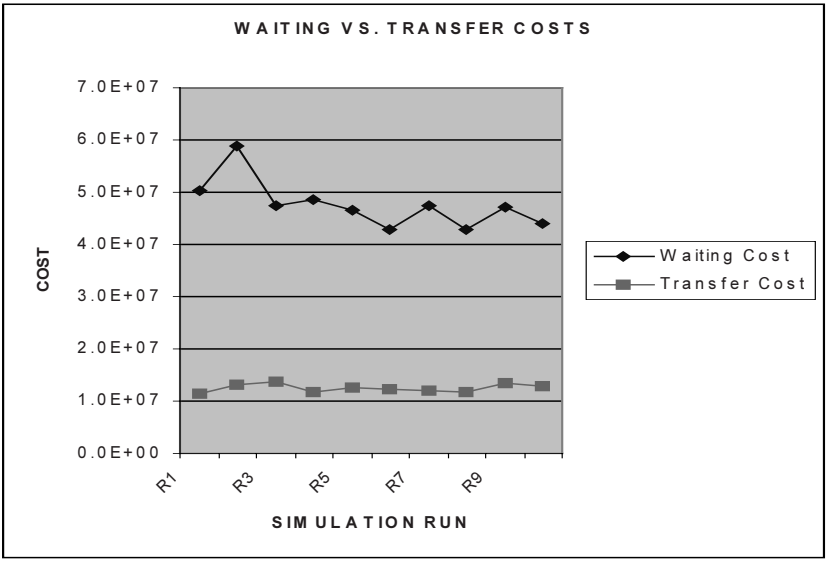


Fig. 3. Transfer cost vs. waiting cost: Transfer cost <<<< waiting cost.

References

1. S.M. Deen and M. Fletcher: "Temperature Equilibrium in Multi-Agent Manufacturing Systems", *Proceedings of the 11th International Conference on Database and Expert Systems Applications (DEXA'00)*, London (2000).
2. R. Axelrod, The Complexity of Cooperation: Agent based Models of Competition and Collaboration, Princeton University Press, New Jersey, 1997. ISBN 0-691-01567-8.
3. S. M. Deen and M. Fletcher: "Fault Tolerance in Holonic Manufacturing Systems", *IEEE International Workshop on Embedded Fault-Tolerant System*s, Boston,USA, (1998).
4. T. Neligwa and S.M. Deen, "A Study of the HMS Operational Architecture", *Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA'02)*, Aix-en-Provence (2002). Pp 608–612.
5. T. Neligwa and M. Fletcher, "An Operational Model for Holonic Manufacturing Systems", in Agent-based Manufacturing: Advances in the Holonic Approach, S.M. Deen (ed), Springer Verlag, Berlin, Germany (2003).
6. R. G. Smith, "The Contract-Net Protocol: high-level communication and control in a distributed problem solver", IEEE Transactions on Computers, **29**(12), 1104–1113 (1980).
7. D.K. Smith, Network Optimisation Practice: A Computational Guide, Ellis Horwood Ltd, West Sussex, England, 1982. ISBN 0-85312-403-5.

Multiagent-Based Process Planning and Scheduling in Context of Supply Chains

Berend Denkena, Michael Zwick, and Peer-Oliver Woelk

Institute of Production Engineering and Machine Tools (IFW), University of Hannover,
Schlosswender Strasse 5, 30159 Hannover, Germany
{denkena, zwick, woelk}@ifw.uni-hannover.de
http://www.ifw.uni-hannover.de

Abstract. Intelligent software agents are a promising approach to improve information logistics in manufacturing enterprises. This paper deals with the application of agents in the area of process planning and production control. Thus, enterprises will be able to fulfil the requirement of flexible, reliable and fault-tolerant manufacturing. Fulfilment of these requirements is a prerequisite for successful participation in modern business alliances like supply chains and virtual enterprises. Thus, agent-based improvements of information logistics enable enterprises to face the challenges of competition successfully. Current research activities focus on the development of agent-based systems for integrated process planning and production control. They led to the "IntaPS" approach, which is presented in this paper. Furthermore, the integration of different agent-based systems in context of collaborative manufacturing in supply chains will be discussed.

1 Introduction

Modern manufacturing is in need of flexible and adaptive concepts for process planning and scheduling to meet market requirements. However, today's industrial products are often characterized by a high complexity of design, functionality and necessary manufacturing and assembly processes. This situation provides the opportunity for small and medium-sized enterprises (SME) to improve their competitiveness within global economy if they are able to adapt to changing environments quickly. They may participate in supply chains and form virtual enterprises to fulfil specific customer demands. Thus, the ability to process necessary information efficiently is of increasing importance, e.g. in case of manufacturing complex and sophisticated products, where process planning and scheduling turn into knowledge-intensive tasks. Thus, so-called "information logistics" is one of the crucial factors for business success of enterprises.

A promising approach to achieve the goal of flexible manufacturing is the application of intelligent agents. They represent a modern area of research in Distributed Artificial Intelligence. Since they are able to collaborate and to solve problems in a distributed manner, they are often used for complex tasks, which can be hardly solved monolithically. In these domains agents gain great benefit. This paper presents the approach to improve in-house information logistics in the field of process planning and scheduling by application of intelligent software agents and integration of in-house agent application into an agent-based supply chain scenario.

V. Ma•ík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 100–109, 2003.

2 Current Situation in the Real World

2.1 Successful Supply Chain Participation Demands Effective Manufacturing

Enterprises have to meet several requirements such as providing a specified product at a defined time to the customer reliably. Especially in supply chains, customers will switch to other contractors for their orders in future if their requirements are not met, e.g. by repeated delivery delay. Thus, unfulfilled requirements will weaken the market position with a lasting influence. From a holistic point of view ("top-down"), supply chains as a whole are as efficient as the weakest "link", respective the most inefficient enterprise. Thus, each enterprise aims to reach a high economic viability and to become a strong, reliable link of the supply chain. Therefore, it is important to take all necessary organisational measures to keep estimated manufacturing costs and due dates as well as to meet contracted product quality. One of these organisational measures deals with improvement of internal information logistics, because availability of information is a crucial factor for modern enterprises.

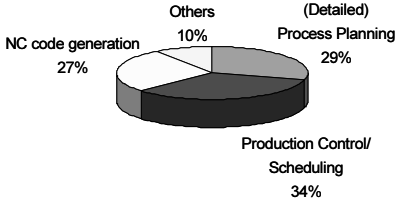
This challenging situation is enforced not only by dynamic behaviour of the supply chain itself but also by other trends in modern product design and manufacturing. It effects internal structures of the enterprise (bottom-up point of view). For example, customers demand highly customised products even in serial production (mass customisation) which leads to a large number of variants. Thus, modern manufacturing systems must handle several products and variants with small lot sizes simultaneously. Furthermore, modern products are characterized by a high complexity of design and take advantage of an integrated design of mechanical, electrical and information processing components (mechatronics). In this context, manufacturing processes need to be improved as well. Thus, improvements of internal information logistics are not only a demand resulting from the holistic point of view of the supply chain as a whole. From a bottom-up point of view, these improvements are necessary for every modern manufacturing system, too.

2.2 Changing Requirements in Process Planning and Shop Control

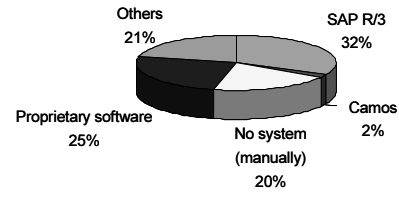
Current changes in the economic environment of enterprises affect the their infrastructure of information technology. Among others, systems used for Computer Aided Process Planning (CAPP) are affected by this trend. Some enterprises try to use integrated enterprise resource planning systems (ERP systems) for this purpose, but most ERP systems are based on centralised system architectures, e.g. SAP R/3, which is used by approximately one third of the companies questioned in a survey described later in this paper. A centralized approach may be suitable for mass production or products with low or medium complexity. In this case, however, straightforward methods for standardised communication meet the moderate demand for co-ordination of the production system. However, these systems may reach their limitations e.g. for the production of goods with increased manufacturing technology requirements and enlarged product complexity combined with small batch sizes and heterogeneous orders. Due to the centralised and monolithic architecture, these systems offer insufficient capabilities for modelling dynamic structures. Nevertheless, modelling dynamic behaviour is necessary for a reliable production and flexible

reaction to internal disturbances (e.g. machine breakdown, missing devices and tool) or external disturbances caused by the market, e.g. changed customer behaviour [1]. Furthermore, structures of companies and production systems are also characterised by "long term" dynamic behaviour.

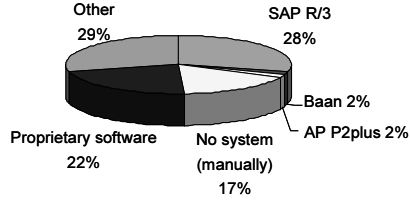
Since it is supposed that decentralised systems are able to handle dynamic behaviour more efficient than centralised approaches, IFW carried out a survey to determine the state of decentralisation in industrial enterprises in Germany. 75 companies from branches like mechanical engineering, automotive industry and supplier for automotive industry (individual or batch-job production mostly) filled out a questionnaire concerning their efforts to decentralise managerial and manufacturing functions and their corresponding IT infrastructure. Approximately a quarter answered at each case: *(a)* "decentralisation is no subject of interest", *(b)* "projects for decentralisation are planned", *(c)* "projects are in progress", and *(d)* "projects had been carried out". Most decentralisation projects aim at reduction of lead time, higher flexibility and robustness or cost reduction. Known obstacles are lack of worker motivation to support decentralisation measures and shortage of manpower for planning and executing decentralisation projects for example. Despite these obstacles, most companies are satisfied with the results of decentralisation: 37% stated that the goal of their projects are fulfilled. Only 3% stated that they missed all of their goals.



a) Shifting of tasks to the shop floor level



b) CAPP systems for process plan generation



c) Systems for production control and scheduling

Fig. 1. Results of a survey on decentralisation in small and medium sized enterprises

The measures taken by the companies deal with shifting tasks to the shop floor (Figure 1a), rearrangement of machines and resources at the shop floor and introduction of team production as well as further education of the workers for example. Another question dealt with the CAPP systems used for process plan generation (Figure 1b). Approximately 32% of the companies use the PP module of the SAP R/3 system for process plan generation, 21% use other commercial systems like "Camos enginObjects" for this purpose. A very notable result is the fact that a quarter of all interviewed companies uses a proprietary software tool for process

planning and 20% perform this task without computer support. A similar situation characterises the IT infrastructure for production control systems (Figure 1c). It is noteworthy that more than one-third of all interviewed companies use proprietary software (22%) or no software (17%) for this purpose. These survey results emphasise the proposition of a lack of appropriate software tools to support necessary tasks like distributed process planning and production control. Thus, the assumption is affirmed that there is a strong need for new information technologies and tools.

3 The Idea of Integrated Process Planning and Scheduling

3.1 Current Problems in the Manufacturing Domain

As mentioned above, two main requirements for future information systems in manufacturing arise: *(a)* these systems must be suitable for implementation in decentralised organisational structures at the shop floor, and *(b)* these systems must provide all necessary services to operate a flexible shop floor with manufacturing of highly customised products. Thus, the ability to perform product-related process planning as well as scheduling and manufacturing execution simultaneously is a crucial factor for future information systems in the manufacturing domain.

Although current requirements demand for a holistic view of process planning and production control, the traditional approach of separating planning from execution results in a gap between the involved systems, which implies loss of time and information. The current situation is characterised by several disadvantages, e.g. conventional static process plans take mainly technological aspects into consideration. However, economical aspects (e.g. capacity, resource load) remain disregarded. Furthermore, urgent process plan modifications, which may become necessary due to unexpected events (e.g. machine breakdown), are carried out at shop floor level, which will lead to feasible, but not to optimal results. Furthermore, complexity of manufacturing processes and knowledge, which is necessary for process planning, increases due to new manufacturing technologies. Despite the fact, that this knowledge is available at shop floor level (e.g. by well trained machine operators), it often takes a long time until it is available at a centralised process planning group. Thus, advantages innovative manufacturing technologies, remain unused.

3.2 Known Approaches to Bridge the Gap

Integration of Process Planning and Production Control

The aim of integration of process planning and production control functionality is well known for several years. Since the end of the 1980ies, several research projects worked on this problem, but most of these projects based on centralised system architectures and used approaches like bulky, non-linear process plans, e.g. the EC joint-research projects FLEXPLAN and COMPLAN [2, 3]. Current research activities use flexible process plans for scheduling of flexible manufacturing systems or apply AI techniques to improve the procedure of process planning [4]. A very interesting approach (“EtoPlan”) is presented by Kals, Zijm and Giebels [5].

Application of Software Agents in the Manufacturing Domain
A very promising approach deals with intelligent software agents, which will improve information logistics in the decentralised manufacturing domain since software agents are suitable to implement distributed systems in particular. Most of current applications of multi-agent based concepts aim at scheduling problems. For example, the "MAPS" system [6] comprises functionality for middle-term and short-term scheduling as well as an interface to communicate with a commercial PPC system ("Production Planning and Control"). Another interesting approach is realised by the "ExPlanTech" system [7], which enhances the approach of performing scheduling with intelligent agents toward aspects of product configuration and inter-enterprise collaboration.

Another important application is the design of "Holonic Manufacturing Systems" (HMS). A "Holon" is an autonomous and co-operative building block of a manufacturing system. It consists of an information processing part, and can also contain a physical processing part or even a human being. Thus, the concept of holons has a wider scope than a pure intelligent software agent. Furthermore, a holon can be part of another holon which leads to the presence of hierarchies (called "holarchies"). An overview of holonic scheduling technologies is given by L. Bongaerts [8].

3.3 Agent-Based Approach of IntaPS

Ongoing research activities at the IFW focus on application of agent technology in the area of process planning and production control in order to bridge the gap in information flow discussed above. These activities led to the IntaPS approach of integrated, agent-based process planning and scheduling. An outline of the IntaPS system architecture is shown in figure 2. It consists of two substantial components, which link together information systems of earlier stages of product development and resources on the shop floor. One component realises a multi-agent system (MAS) enabling decentralised planning on shop floor level, the other component performs rough level process planning tasks.

Decentralised Components on Shop Floor Level
The implementation of decentralized planning unit on the shop-floor level is based on three different types of agents: resource agents, order agents, and service agents. Each relevant resource of the production system (machine tools, transportation devices, staff, virtual resources like information systems, etc.) and its environment is represented by a *resource agent*. Resource agents provide local knowledge bases of the associated resources. Furthermore, legacy information systems (e.g. CAM systems for NC code generation) may be integrated into the system architecture using resource agents "wrapping" these information sources.

Order agents are representing orders which have to be manufactured. Due to its autonomy and pro-activity, an order agent is able to recognize internal and external disturbances and to react appropriately. Order agents optimise their behaviour using utility functions representing their individual goals. Different order agents vary in divergent weights of goals within their utility functions: An order agent representing a rush order will rate the goal "finished on scheduled due date" higher than the goal "using cost-efficient manufacturing processes", for example. An order agent

representing a stock order should prefer the opposite weight of goals. Since lead times in production of high-sophisticated products may be very long compared to mass production and boundary conditions may change significantly, orders are represented by agents, not as passive objects resources have to work on. *Service agents* are used for human interaction, transparency and maintenance purposes.

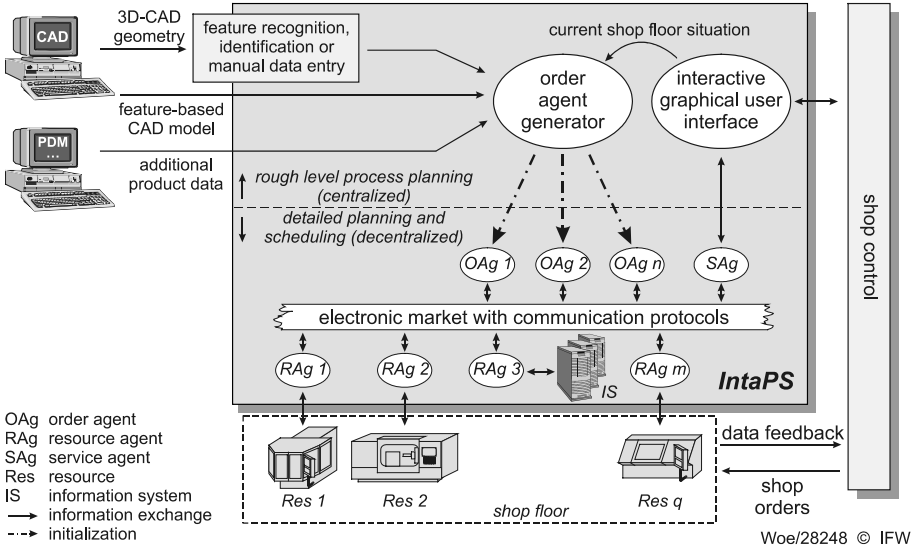


Fig. 2. Basic Architecture of the IntaPS Approach

The detailed process planning and scheduling takes place co-operatively within an *electronic marketplace*. Order agents and resource agents interact according to a three-phase model. Starting with a "negotiation phase" required manufacturing skills and due dates as well as capabilities and capacities are communicated. Suitable sequences of manufacturing operations result from auctions between appropriate partners. The optimal sequence of operations is accepted as detailed plan. The second phase is called "verification phase" and ensures the feasibility of the detailed plan. The order agent examines continuously whether its detailed plan is executable under the current conditions. Changing situations (e.g. machine breakdown) cause agents to analyse the consequences and to identify those parts of their detailed plans which are affected. If necessary, the order agent enters a "re-negotiation phase" and tenders parts of the detailed plan for a new auction. The re-negotiation phase leads to an improved alternative detailed plan which substitutes the previous plan. Afterwards the verification phase is resumed and lasts until the order is finished.

Centralised Rough-Level Process Planning Component

Centralized rough level planning pre-processes incoming data and generates a rough level process plan. These data are geometrical or technological information about the product (e.g. from CAD systems), further organisational information related to products and orders (e.g. from PDM or ERP systems) as well as information concerning the current shop floor situation. Order agents are synthesised with respect

to these information and initialised with a rough level process plan. This plan contains information only, which the agent is not able to recognise from the environment by itself (e.g. constraints between manufacturing operations). Thus, order agents obtain a maximum scope for allocation of suitable resources and time slots for manufacturing. Since product-related information are extracted from PDM systems and contained in the rough-level process plan and products are not regarded as active part from point of view from the shop floor, the IntaPS architecture is not in need of a specific product agent like used in PROSA-based systems. Nevertheless, IntaPS can be integrated in holonic systems (e.g. for supply chain management) using a PROSA architecture and inquire product-related information from product holons instead of PDM systems. Furthermore, the centralized part of the IntaPS architecture provides a graphical user interface for interaction with the system.

Distribution of Tasks between Centralised and Decentralised Components
Centralized CAPP approaches perform all necessary tasks for process plan generation at a single location (e.g. in a single instance of the CAPP program). In distributed system architectures these tasks should be distributed among the involved software components. Since the benefit of multi-agent systems results from emergent behaviour through efficient communication, the strategy for system design should be to make the single agent as complex as necessary but also as simple as possible. The following table lists typical activities which have to be carried out in process planning. The abbreviation behind each activity names the responsible entity:

1. Determination/design of raw part	RLPP
2. Identification of machining tasks	RLPP
3. Divisions/sequencing of machining operations	(RLPP /) OA – RA
4. Selection of machine tool	OA – RA
5. Selection of cutting tools	RA
6. Selection of machining parameters	RA
7. Estimation of time and cost	RA
8. Generation of NC code	*
9. Documentation of process plan	OA – RA / SA

RLPP stands for "Rough-level Process Planning" carried out by the centralised component (e.g. analysing CAD models of the product and manufacturing feature identification based on ISO 14649 "STEP-NC"). Sequence of machining operations and selection of machine tools are results of negotiation between order agents and resource agents (*OA – RA*). Known restrictions for sequencing are implied in the formal description of identified machining tasks generated by the RLPP (indicated by *(RLPP/)*). While calculating bids and offers during negotiation, resource agents evaluate their ability to perform requested machining tasks (indicated by *RA*). Thus, complete process plans (from the point of view of orders) and loading status (from the point of view of resources) are result of negotiation. Since human users of planning systems like IntaPS like to have a "bird-eyes view" over the shop floor situation in a whole, service agents (*SA*) are used to gather information and to visualise them at the user interface at users request. The activity of NC code generation is marked with an asterisk (*), since the performing entity depends on the organization of the individual company. If the company uses shop-floor-oriented programming procedures (SOP),

the respective functionality has to be integrated into the resource agents. If a centralized CAM department generates NC code, this department itself may be represented by a resource agent and other resource agents use its offered services. Some additional tasks performed by process planning groups in enterprises are not covered by functionality of the IntaPS system, e.g. processing bill of materials (map design part list to manufacture/assembly bill of material), inspection planning (while inspection stations are not represented yet by resource agents), and consulting service for product design group to achieve a proper "design-for-manufacturing". Nevertheless, the planner will be able to spend more time on these additional tasks due to time-saving generation of process plans and less efforts for manual interaction in case of re-planning.

4 IntaPS Approach on Its Way to the Real World

4.1 Prototype Implementation and Current Developments

At the moment, the JAVA-based IntaPS prototype implementation are realised using the FIPA-compliant 'JADE' agent platform [9]. In addition to standardised components of the agent platform, further enhancements (e.g. adaptive communication protocols and knowledge representation) are part of the IntaPS project [10]. Nevertheless, the prototype system is still under development. Current enhancements deal with topics to improve suitability of the approach for real-world manufacturing systems mostly. One of these topics is the introduction of hierarchical structures in order to realise semi-autonomous production structures like manufacturing islands (real as well as virtual). Currently, IntaPS is using a resource-oriented shop floor model where negotiations only take place between resource agents and order agents directly. In future, order agents will negotiate with resource agents representing groups of machines (manufacturing islands). Co-ordination within these groups may realised by additional multi-agent systems. In this context, IntaPS will gain benefit from holonic concepts of hierarchical production structures. Furthermore, hierarchical structures will be used for representation of more complex products and assemblies as well. Handling hierarchical structures enables not only more detailed modelling of cooperative manufacturing but also integration of the (intra-enterprise) prototype system into more complex (inter-enterprise) supply chain scenarios.

4.2 Realising *Agent.Enterprise*

An inter-enterprise supply chain scenario like mentioned above is *Agent.Enterprise*. The *Agent.Enterprise* scenario is an initiative of the special interest group on "Information Systems in Manufacturing Logistics" (SIG Manufacturing Logistics) of the German priority research program 1083 "Intelligent Agents in Real-world Business Applications". The IntaPS project is part of this research program and involved into development of *Agent.Enterprise* together with four other research projects [11]. Two of these research projects focus on supply chain aspects: SCM scheduling (DISPOWEB project) as well as tracking and tracing of supply chains

(ATT project). Three projects deal with application of agent technology in the participating enterprises of the supply chain with a special focus on integration of process planning (IntaPS project), reliability and robustness (KRASH project), and batch production of semiconductors (FABMAS project).

Each project implemented a prototype multi-agent system to conduct specific tasks in order to fulfil individual project requirements. Furthermore, necessary processes in supply chains were analysed and modelled. Thus, interfaces were designed between involved systems: DISPOWEB receives customer requests and performs planning and scheduling on supply chain level based on status information enquired from MAS on enterprise level (represented by IntaPS, KRASH, and FABMAS respectively). MAS on enterprise level simulate execution of customer orders while these orders are tracked by ATT. Each involved MAS has an interface agent which is able to communicate with agents of its own MAS as well as other interface agents within *Agent.Enterprise*. Communication between interface agents is based on (sometimes iterated) sequences of standard FIPA protocols and a supply chain ontology tailored for this scenario [12]. Since the MAS reside on agents platforms all across Germany (Bremen, Frankfurt, Karlsruhe, Ilmenau, and Erlangen), *Agent.Enterprise* utilises the AgentCities infrastructure including directory facilitator and exchange of FIPA ACL messages using the HTTP MTP. Current developments are dealing with enhancements in synchronisation of distributed MAS and interfaces to further MAS in order to make *Agent.Enterprise* an open platform e.g. to test MAS in manufacturing as part of the AgentCities network.

5 Summary

This paper presents the approach of integrated agent-based process planning and scheduling to bridge the gap between planning and execution on shop floor level. Thus, enterprises will be enabled to perform more flexible and robust manufacturing processes. Furthermore, improved information logistics and application of decentralised approaches like multi-agent systems in IT infrastructure fit to today's requirements of decentralised organisational structures in modern enterprises. Due to the approach of IntaPS, capacity information and due dates will be taken into consideration for early stages of process planning. On the other hand, process planning knowledge will be used for short term scheduling decisions at the shop floor. Therefore, problems will be eliminated which result from time-delayed return of manufacturing knowledge and capacity data or other lacks of information flows e.g. from the use of static process plans. The presented approach of the IntaPS project and its prototype implementation contributes to these ongoing developments. In addition, IntaPS is involved in a joint research program building up the *Agent.Enterprise* scenario. *Agent.Enterprise* brings together multi-agent system on enterprise level as well as on supply chain level. Since these multi-agent systems are able to interact along the whole supply chain, *Agent.Enterprise* will point out the feasibility of agent technology in large-scale scenarios.

Acknowledgement. The presented work results from the IntaPS research project, funded by the Deutsche Forschungsgemeinschaft (DFG) within the project To 56/149 as part of the priority research program 1083 "Intelligent Agents and Real-World Business Applications". IntaPS is carried out together with the Center of Computing Technologies (TZI), University of Bremen. Further information is available at http://www.intaps.org .

References

1. Tönshoff, H.K., Teunis, G.: Modular Shop Control Toolkit for Flexible Manufacturing. Production Engineering **2** (1998) 111–114
2. Tönshoff, H.K., Beckendorff, U., Andres, N.: FLEXPLAN – A Concept for Intelligent Process Planning and Scheduling. Proc. of the CIRP Int. Workshop, Hannover (1989).
3. Kruth, J.P., Detand, J.: CAPP system for nonlinear process plans. Annals of the CIRP **1** (1992) 489–492
4. Teti, R., Kumara, S.R.T.: Intelligent Computing Methods for Manufacturing Systems. Annals of the CIRP **2** (1997) 629–652
5. Giebels, M.: EtoPlan – a Concept for Concurrent Manufacturing Planning and Control. University of Twente, Netherlands (2000)
6. Wellner, J., Dilger, W.: MAPS – A Multi-Agent Production Planning System. ". Workshop on Intelligent Agents in Information and Process Management, 22nd German Annual Conference on Artificial Intelligence, Bremen (1998) 71–78
7. Říha, A., Pechoucek, M., Vokrínek, J., Maník, V.: ExPlanTech: Exploitation of Agent Technology in Production Planning. Lecture Notes in Artificial Intelligence, Vol. 2322. Springer Verlag, Berlin Heidelberg New York (2001)
8. Bongaerts, L.: Integration of Scheduling and Control in Holonic Manufacturing Systems. Katholieke Universiteit Leuven, Belgium (1998)
9. Bellifemine, F., Poggi, A., Rimassa, G.: JADE – A FIPA-compliant agent framework. Proc. of PAAM'99, London, UK (1999) 97–108
10. Timm, I.J., Tönshoff, H.K., Herzog, O., Woelk, P.-O.: Synthesis and Adaptation of Multiagent Communication Protocols in the Production Engineering Domain. Proc. of 3rd Int. Workshop on Emergent Synthesis, Bled, Slovenia (2001) 73–82
11. Woelk, P.-O. *et al.* Agent.Enterprise. Internet http://www.realagents.org/geturl.php?ID=1010 (Accessed 2003-03-20)
12. Frey, D., Stockheim, T., Woelk, P.-O., Zimmermann, R.: Integrated Multiagent-based Supply Chain Management. 1st Int. Workshop on Agent-based Computing for Enterprise Collaboration, Linz, Austria (2003, *accepted for publication*)

Improving Multi-agent Based Scheduling by Neurodynamic Programming

Balázs Csanád Csáji, Botond Kádár, and László Monostori

Computer and Automation Research Institute,
Hungarian Academy of Sciences, Kende u. 13-17
Budapest, H-1111, Hungary
{csaji, kadar, laszlo.monostori}@sztaki.hu

Abstract. Scheduling problems, e.g., a job-shop scheduling, are classical NP-hard problems. In the paper a two-level adaptation method is proposed to solve the scheduling problem in a dynamically changing and uncertain environment. It is applied to the heterarchical multi-agent architecture developed by Valckenaers et al. Their work is improved by applying machine learning techniques, such as: neurodynamic programming (reinforcement learning + neural networks) and simulated annealing. The paper focuses on manufacturing control, however, a lot of these ideas can be applied to other kinds of decision-making, as well.

1 Introduction

Continuous, steady improvement is a key requirement for manufacturing enterprises that necessitates flat and flexible organizations, life-long learning of employees on the one hand, and information and material processing systems with adaptive, learning abilities on the other hand [11]. The paper outlines an attempt to enhance the performance of an agent-based manufacturing system by using adaptation and machine learning techniques. A two-level adaptation method is proposed to solve the scheduling problem in a dynamically changing and uncertain environment. It is applied to the heterarchical multi-agent architecture developed by Valckenaers et al. [19], which was inspired by food forging ants. First, the general job-shop scheduling problem is described and its complexity is highlighted. A short introduction to the advantages of multi-agent systems is given and the PROSA architecture overviewed including its improvement with mobile agents (ants). Some properties of neurodynamic programming are also described and, finally, an adaptation method is presented which was applied to a multi-agent system. It is capable of solving the job-shop scheduling efficiently and with great fault tolerance. The applicability and the effectiveness of the proposed method are illustrated by the results of experimental runs.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 110–123, 2003.

2 Scheduling

Scheduling is the allocation of resources over time to perform a collection of jobs or tasks. The problem of scheduling is important in manufacturing: near-optimal scheduling is a pre-requisite for the efficient utilization of resources and hence, for the profitability of the enterprise. Moreover, much of what we can learn about scheduling can be applied to other kinds of decision-making and therefore, is of general practical value.

2.1 Job-Shop Scheduling

One of the basic scheduling problems is the so-called job-shop scheduling. We begin by defining the general job-shop problem: suppose that we have n jobs $J = \{J_1, J_2, ..., J_n\}$ to be processed through m machines $M = \{M_1, M_2, ..., M_m\}$. The processing of a job on a machine is called operation. The operations are non-preemptive (they may not be interrupted) and each machine can process at most one operation at a time (capacity constraint). Each job may be processed by at most one machine at a time (disjunctive constraint). O denotes the set of all operations. Every job has a set of operation sequences: the possible process plans, $o: J \rightarrow P(O^*)$. These sequences give the precedence constraints of the operations. The processing time of an operation on a machine is given by $p: M \times O \rightarrow R^+$ which is a partial function. Due dates are given for every job: $d: J \rightarrow R^+$, $d(J_i)$ is the time by which we would like to have J_i completed ideally. It is not easy to state our objectives in scheduling. They are complex, and often conflicting. Generally, we can say that the objective is to produce a schedule that minimizes (or maximizes) a performance measure f, which is usually a function of job completion times (e.g.: maximum completion time, mean flow time, mean tardiness, number of tardy jobs, etc.). Therefore the job-shop scheduling is an optimization problem.

2.2 Complexity of Scheduling

The general job-shop scheduling is a very complex problem. If we suppose, e.g., that every job consists of exactly m operations and each machine can do every operation, then the size of the search space is $(n!)^m$. Thus, it is not possible to try every potential solution, not even in cases such as $n = 20$ and $m = 10$ because even in this case the size of the search space is much larger than the number of particles in the known Universe[1]. Except for some strongly restricted special cases[2], the job-shop scheduling is an *NP-hard* optimization problem [14]. It means that no polynomial time algorithm

---

[1] According to Arthur Eddington (*Mathematical Theory of Relativity*) the number of particles in the known universe is $\approx 3.1495 \cdot 10^{79}$ and $(20!)^{10} \approx 7.2651 \cdot 10^{183}$.

[2] Like single or double machine makespan / maximum lateness problems.

exists that always gives us the *exact* optimal schedule, unless[3] $P = NP$. Moreover, there is no good polynomial time approximation of the optimal scheduling algorithm: Williamson et al. [22] gave the first nontrivial theoretical evidence that shop-scheduling problems are hard to solve even approximately. They showed that if f (the performance measure) is the maximum completion time ($C_{\max}$), then if a (polynomial time) ε–approximator with $\varepsilon < 5/4$ exists for the job-shop problem then it implies that $P = NP$. Because of these properties, the job-shop scheduling has earned the reputation of being notoriously difficult to solve.

3 Multi-agent Systems

Many different approaches such as *integer programming* [15], *Lagrangian relaxation* [4], *branch and bound algorithms* [13] [2], *simulated annealing* [10], *genetic algorithms* [3] [6], *neural networks* [9], *reinforcement learning* [23], etc. have been tried for solving the job-shop scheduling problem. Though, some of them, e.g., integer programming, have elegant mathematics they scale exponentially and are only able to solve highly simplified "toy" instances. Others, such as genetic algorithms, have some promising experimental results but it is not clear how to put these approaches into a distributed (multi-agent) system. As we stated before, our objective is to propose an adaptive extension to a previously designed heterarchical multi-agent architecture. Before we continue our investigation on job-shop scheduling, let us give a short review on multi-agent systems in general and in manufacturing. According to Baker [1], an agent is basically a self-directed software object. It is an object with its own value system and a means to communicate with other objects like this. Unlike a lot of software, which must be specifically called upon to act, the agent software continuously acts on its own initiative. In a heterarchical architecture, agents communicate as peers, no fixed master/slave relationships exist, each type of agents is usually replicated many times, and global information is eliminated. Advantages of these heterarchical multi-agent systems include: *self-configuration*, *scalability*, *fault tolerance*, *emergent behaviour*, and *massive parallelism* [1]. Other authors claim that the advantages of heterarchical architectures also include *reduced complexity*, *increased flexibility*, and *reduced cost* [20], [7], [5], [16], [18] as well. This approach is useful for manufacturers who often need to change the configuration of their factories by adding or removing machines, workers, product lines, manufacturers who cannot predict the possible manufacturing scenarios according to which they will need to work in the future.

3.1 The PROSA Architecture

PROSA, developed by Van Brussel et al. [21], is a *holonic* reference architecture for manufacturing systems. It is a starting point for the design and development of multi-agent manufacturing control. The structure of the architecture identifies three kinds of

[3] Which is the largest unsolved problem in complexity theory, however we have a strong intuition that the two sets are different.

agents (holons), their responsibilities, and the way they interact. The basic architecture consists of three types of basic agents: order agents (internal logistics), product agents (process plans), and resource agents (resource handling). *Resource agents* correspond to physical parts (production resources in the manufacturing system, like: factories, shops, machines, furnaces, conveyors, pipelines, material storages, personnel, etc.), and contain an information processing part that controls the resource. *Product agents* include the process and product knowledge to ensure the correct production. They act as information servers to other agents. *Order agents* represent a task or job in the manufacturing system. They are responsible for performing the assigned work correctly and on time.

3.2 Ant Colonies

Many approaches in multi-agent systems were inspired by heterarchical biological systems such as wasp nests, bird flocks, fish schools, wolf packs, termite hills and ant colonies. Valckenaers et al. [19] presented a coordination and control technique, which was inspired by food-forging ants. They identified the key achievement of this biological example: limited exposure of the individuals, combined with the emergence of robust and optimized overall system behavior. In their work the environment was agentified and made it part of the solution. The PROSA reference architecture was applied to separate resource, logistic, and process concerns and new type of agents were introduced, called ants, which are mobile and they gather and distribute information in the manufacturing system. Their main assumption is that the agents are much faster than the ironware that they control, and that makes the system capable for forecasting. Agents are faster and, therefore, can emulate the system's behavior several times before the actual decision is made. Scheduling in this system is done based on local decisions. Each order agent sends ants moving downstream in a virtual manner. They gather information about the possible schedules from the resource agents and than they return to the order agent with the information. The order agent chooses a schedule and sends ants to book the needed resources. After that the order agent regularly sends ants to rebook the previously found best schedule, because if the booking is not refreshed, it *evaporates* (like the pheromone in the analogy of food-forging ants) after a while. From time to time, the order agent sends ants to survey the possible new (and better) schedules. If they find a better solution, the order agent books the resources that are needed for the new schedule and the old booking information simply *evaporates*.

3.3 Adaptive Approach

The multi-agent manufacturing system inspired by food-forging ants is very promising, but as to the scheduling problem it can be regarded as a framework only. From what we stated above on the complexity of scheduling, it is clear that even if the mobile agents are much faster than the ironware, they cannot survey every possible schedule. They should select schedules from an initial set for investigation. They

should adapt to the current state of the system and make guesses about the presumably good schedules. If the system changes, they should *explore* the new situation. If the system stabilizes, they should *exploit* the information they have gathered. In the paper we propose a two-level adaptation mechanism with these properties. Our below approach applies *neurodynamic programming* (*reinforcement learning + neural networks*) and *simulated annealing*.

4 Neurodynamic Programming

Two main paradigms of machine learning are known: learning with a teacher, which is called *supervised learning*, and *learning without a teacher*. The paradigm of learning without a teacher is subdivided into *self-organised (unsupervised) learning* and *reinforcement learning*. Supervised learning is a "cognitive" learning method performed with the support of a teacher: this requires the availability of an adequate set of input-output examples. In the contrary, reinforcement learning is a "behavioral" learning method, which is performed through *interaction* between the learning system and its environment. The fact that this interaction proceeds without a teacher makes reinforcement learning particularly attractive for dynamic situations. We refer to the modern approach of reinforcement learning as *neurodynamic programming*, because dynamic programming provides its theoretical foundation and neural networks provide its learning capability. The operation of a reinforcement learning system is characterized as follows [8]:

1. The environment evolves by probabilistically occupying a finite set of discrete states.
2. For each state there is a finite set of possible actions that may be taken.
3. Every time the learning system takes an action, a certain reward is incurred.
4. Sates are observed, actions are taken, and rewards are incurred at discrete time steps.

The agents' goal is to maximize their *cumulative profit* (reward). This does not mean maximizing immediate gains, but the profit in the long run. In our scheduling system the rewards are computed from the performance measures of the achieved schedules.

4.1 Markov Property

An important concept in reinforcement learning is the *Markov property*. The environment satisfies the Markov property if its state signal compactly summarizes the past without degrading the ability to predict the future. Formally, if we denote the state at time t by s_t, the action taken by a_t and the reward received by r_t, we can compute how the environment might response at time $t+1$ to the action taken at time t. If

the system has Markov property, the response of the environment at $t+1$ depends only on the state and action representations at t. A state signal has Markov property if and only if:

$$P(s_{t+1} = s, r_{t+1} = r \mid s_t, a_t) = P(s_{t+1} = s, r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, s_0, a_0)$$

for all s, r, s_t, a_t and for all histories s_t, a_t, r_t, …., r_1, s_0, a_0. In this case the environment and the task as a whole are also said to have the Markov property [17]. A reinforcement learning task that satisfies the Markov property is called a *Markov Decision Process (MDP)*. In our work we presuppose that our system states satisfy the Markov property. We may safely assume so, because the only thing that matters if we want to develop an incremental scheduling algorithm, is the current schedule of the resources. It is not relevant how this schedule has evolved.

4.2 Temporal Difference Learning

Reinforcement learning methods are based on a *policy* π for selecting actions in the problem space. The policy defines the actions to be performed in each state. Formally, a policy is π: $S \times A \rightarrow [0,1]$ a mapping (partial function) from state and actions to the probability $\pi(s, a)$ of taking action a in state s. A *value* of a state s under a policy π is the expected return when starting in s and following π, thereafter,

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\},$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called *discount rate*. (If $\gamma = 0$, the agent is "myopic", and as γ approaches 1, the agent becomes more farsighted.) As in most reinforcement learning work, we attempt to learn the value function of the optimal policy π^*, denoted by V^* rather than directly learning π^*:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

To learn the value function we can apply the method of *temporal difference learning* known as $TD(\lambda)$, developed by Sutton [17]. A value function $V^{\pi}(s)$ is represented by a function approximator $f(s, w)$, where w is a vector that holds the parameters of the approximation e.g., weights if we use neural networks. If the policy π were fixed, $TD(\lambda)$ could be applied to learn the value function V^{π} as follows. At step $t+1$, we can compute the temporal difference error at step t as:

$$\delta_t = r_{t+1} + \gamma \cdot f(s_{t+1}, w) - f(s_t, w)$$

Then we compute the smoothed gradient:

$$e_t = \nabla_w f(s_t, w) + \lambda e_{t-1}$$

Finally, we can update the parameters according to:

$$\Delta w = \alpha \delta_t e_t,$$

where λ is a smoothing parameter that combines the previous gradients with the current gradient e_t, and α is the learning rate. This way, $TD(\lambda)$ could learn the value function of a *fixed* policy. But we want to learn the value function of the *optimal* policy. Fortunately, we can do this by the method of *value iteration*. During the learning we continually choose an action that maximizes the predicted value of the resulting state (with one step look ahead). After applying this action, we get a reward, and update our value function estimation. This means that the policy continuously changes during the learning process. $TD(\lambda)$ still converges under these conditions [17].

5 Adaptive Scheduling

Now, we can return to the problem of scheduling and present our approach. As we stated before, we propose a two-level adaptation improvement to the framework designed by Valckenaers et al. [19]. Intuitively, the function of the first level is to *route* the mobile agents (ants) so they should not have to investigate every possible schedule. Some presumably good schedules are selected on the basis of information gathered previously. The second level of adaptation controls how *greedy* the system is, more precisely, the ratio of *exploitation* and *exploration*.

5.1 Markov Decision Process

First, we investigate the first level of adaptation. Recall that the order agents are responsible for the processing of the jobs and the resource agents control the ironware. When an order agent sends mobile agents (ants) to gather information from the resource agents about the feasible schedules, they cannot investigate every possible schedule (see the part about the complexity of scheduling) and, therefore, when they reach a decision point, they have to choose only *some* possibilities but they should not choose *all* of them except in case of problems of very small size. We suggest that they use the information gathered by the resource agents. The resource agents can learn the possible good schedules of the actual state of the system and only send ants in directions which are presumably good. Every resource agent can learn how to direct ants passing it. To learn the promising scheduling traces we use temporal difference learning. State s that we use for deciding which action is to be taken is the remaining operation sequence of the job that the ant investigates with the earliest start time. Consequently, the state space is $O^* \times R$. A possible action a is the sending of the ant to a resource agent whose machine can process the next operation of the job (the

first operation of the remaining operation sequence). Note that it is possible to send an ant to the sender resource agent. $\pi(s, a)$ is the probability of taking action a in the state s if we use policy π. The randomization of ant routing is not only needed for exploration but it also helps us to avoid pathologic behavior.

A difference form the "classical" Markov Decision Processes is that a resource agent is not restricted to send the ants in one direction only. We threat every $\pi(s, a)$ as independent probabilities and it is possible to take two or more actions in a state. This is called *branching*. Thus, we can explore several schedules. Naturally, we have to be very careful about the branching factor we use, otherwise, the system could turn intractable. This way, the ants move from agent to agent and sometimes they branch. When an ant virtually investigated a schedule (its remaining operation sequence is empty), it computes the performance measure of the achieved schedule and then it starts traveling back to the resource agents and the order agent that started it to support feedback. The feedback information is communicated during the back traveling process. The rewards for the $TD(\lambda)$ reinforcement learning mechanism are computed form the achieved performance measures. If an ant arrives back at a resource agent where there was previously a branching, then it waits until all of the other ants arrive back, and than only the ant with the best schedule travels back to the previous resource agent, the others terminate (let us call this *unbranching*). Every time an ant arrives back at a resource agent, it gives a reward according to the schedule that was investigated. The resource agent uses this information to learn the optimal scheduling routes of the current system. Similarly to the authors of [19] we, too, assume that the agents work much faster than the ironware they control, so they can repeat this process several times before the system changes. Thus, from the viewpoint of the agents the system changes very slowly.

5.2 Simulated Annealing

At this point two questions arise, namely, how we should set the branching factor of ants and what happens if the system changes? These questions are answered on the second level of adaptation. A *simulated annealing* mechanism [12] controls the "temperature" of the system, which is the expected branching number of ants at a resource agent. To keep things as simple as possible, let us suppose that this branching factor is equal regarding all of the resource agents. Consequently, we can define the "temperature" by $T = E\{\pi(s, a)\}$ for a state s. Note that the expected number of branches is not necessarily an integer. If the system changes, then we raise the temperate (the expected number of branches) to force the system to *explore* the new situation. If the system stabilizes, we slowly cool the system down by lowering the temperature to achieve that the agents *exploit* the information they have gathered. We can change the temperature by simply rescaling the probabilities.

Although it is a difficult question in itself and investigating that was not the main goal of our research, some ideas are provided on how the system has been changed. If an order agent books a new schedule, the situation changes, but in that case the order agent can inform the system (the other order agents) about it. Machine(s) may break down or a new machine(s) may become available, but in these cases the resource

agents, which control the changed machines, should inform the system about the change(s). Because our learning algorithm can work in relatively slowly changing non-stationary environments, not recognizing system changes does not disastrously effect the whole system. But if we can recognize a change, we can force the system to make more exploration.

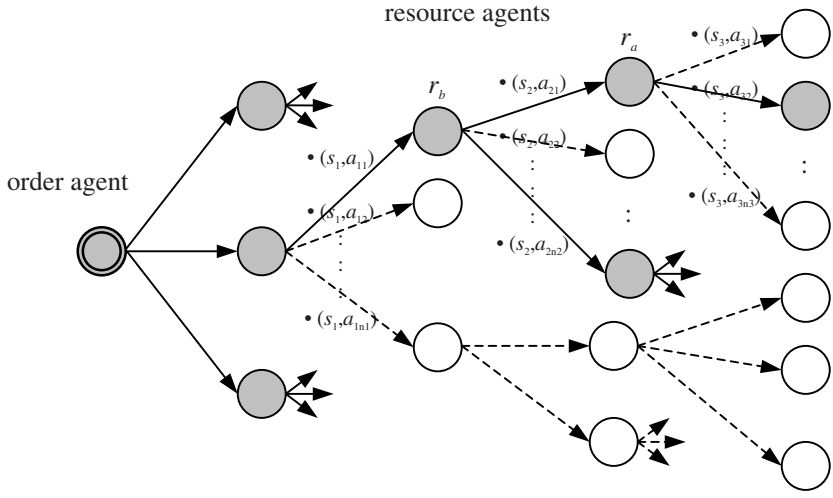


Fig. 1. This figure shows a possible way of ants in the system. They all start from an order agent and they visit resource agents only which are colored gray. At an agent they continue their way in a direction with probability $\pi(s, a)$. Resource agent r_a does a normal routing but at r_b there is a branching

6 Experimental Results

In order to verify the above algorithm, experiments were initiated and carried out. Although the evaluation and analysis of this method is far from being over, some preliminary results are presented here. In the test program, the aim of scheduling was to minimize the maximum completion time (C_{max}). We have extended the job-shop scheduling problem by considering also the distances between machines (note that the distance between two machines can be infinite as well). We did not use any function approximator in our program, because the state space (of reinforcement learning) was relatively small, however, for large problems one cannot avoid using an approximator (e.g., neural networks are candidates for such an approximator). The methodology of testing was as follows: first, we made an advance schedule by scheduling random orders with simple dispatching rules. We did it to generate a non-empty schedule. Then we generated order agents to given jobs and tested how quickly they can find a

good (or the optimal) schedule with different parameter values. We also compared our results with other scheduling algorithms (such as branch and bound). We have tested the exploration / exploitation features of our algorithm, as well. Since our solution is a randomized algorithm, we have generated our results, which we present here, by averaging several (usually a hundred) runtime outcomes. We also give the (standard) deviation of our samples.

Table 1. This table shows a comparison between different scheduling algorithms. It shows how many steps they required (step = virtually putting an operation to a machine) and the performance that they achieved. The BF is the "Brute Force" algorithm (it tries every possible variation). It is only shown because from that column, one can see the size of search space and the optimal performance. The BB column shows the data for the "Branch and Bound" algorithm. It always finds the optimal schedule. The ND(X) columns show the data for our "Neurodynamic" algorithm with branching factor X. The number of iterations is also shown. We have generated the data for our algorithm by averaging several runtime results (for each type of branching factor)

	BF	BB	ND(2.0)	ND(2.0)	ND(2.0)	ND(3.0)	ND(4.0)
Iterations:	-	-	10	50	100	35	25
Steps:	$1.5 \cdot 10^{19}$	$5.6 \cdot 10^{6}$	$1.2 \cdot 10^{4}$	$1.3 \cdot 10^{5}$	$3.8 \cdot 10^{5}$	$5.4 \cdot 10^{5}$	$1.5 \cdot 10^{6}$
Sample Size:	-	-	100	100	100	100	100
(Mean) Performance:	262	262	291.5	268.3	267.1	263.4	262.5
(Standard) Deviation:	-	-	14.36	7.43	7.21	3.19	2.73

While there are many ways to compute decision probabilities from the previously learnt return estimations, in our test program we used a particular one, called *Boltzmann distribution*. We modified the original formula to let the system have higher (than one) expected branching numbers. Here, we briefly present, how we computed the probabilities of sending ants from the learnt value estimations: suppose that a resource agent r has estimation for the expected return value if it sends an ant with the remaining operation sequence γ to the resource agent a at time t. Let us denote that by $V_a(\gamma, t)$. Let us denote the branching factor by β. If our aim is to *maximize* the achieved return values, we can compute the probabilities of sending ants from these data in the following way (note that this formula is slightly more complicated when we consider distances as well):

$$P(\text{sending an ant to } a) = \min\left\{1, \quad \frac{e^{V_a(\gamma,t)/\tau}}{\sum_b e^{V_b(\gamma,t)/\tau}}\beta\right\}, \tag{1}$$

where τ is the so-called *Boltzmann temperature*. High temperatures cause the actions to be all (nearly) equiprobable. Low temperature causes a greater difference in selection probability for actions that differ in their value estimation. As we can see, high Boltzmann temperature also causes the system to make more explorations. The formula that we should use if we want to *minimize* the achieved return values, like in the case of minimizing C_{max}, can be easily computed from (1). As we stated before, in our test program we also consider distances (transportation time) as well between machines. One can further modify the Boltzmann distribution by adding transportation time to the formula. If we denote the transportation time between the resource r and a by $d(r, a)$, we can modify the formula (1) by adding that value to the time parameter. Thus, $V_a(\gamma, t)$ will be $V_a(\gamma, t + d(r, a))$. One of our future plans is to schedule transportation resources too (such as AGVs, trolleys, conveyor belts). Considering distances between resources (machines) is a step into that direction.

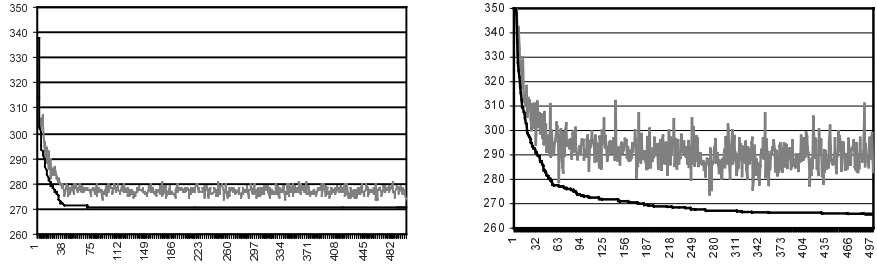


Fig. 2. This figure shows a comparison between different Boltzmann temperatures. Axis x shows the number of iterations. The black line (down) shows the best-achieved performance measure, the gray one (up) the actually achieved performance. The algorithm showed on the left-hand side used 0.3 as Boltzmann temperature. That made the algorithm converge faster, but the probability of choking in a local minimum is also higher. The algorithm on the right-hand side had 0.9 as Boltzmann temperature and that made it take more risk (it made more explorations). It converged slower but it did not choked in local minimums. (Both algorithms had 2.0 as branching factor and 0.1 as learning rate.) We have generated these figures by averaging results of twenty runs

7 Concluding Remarks

In the paper a two-level adaptation mechanism was presented to improve the performance of a previously developed multi-agent based manufacturing control system. The system can learn the routing of mobile agents, called ants, which gather information about the possible schedules of a particular job. In the proposed system the resource agents route the ants. Temporal difference learning is used for learning the directions (probabilities) of the possible good schedules. The branching factor of ants is controlled by simulated annealing. The main advantages of the algorithm are as follows:

1. *Parallel*: the computation of schedule is massively parallel: all of the order and resource agents work independently.
2. *Robust*: the system can handle changes: for example, the constant learning rate makes the system "forget" old information. Using neural networks as function approximator also makes the system more stable, because one of the well-known properties of neural networks is that they have great fault tolerance.
3. *Iterative*: the algorithm can support us (sub-optimal) solutions in settable time slices. It permanently tries to improve the previously found best schedule.
4. *Flexible*: one can change the working of the algorithm by changing the *branching temperature* or the *Boltzmann temperature* or even the *learning rate*.
5. *Fast convergence*: the solution statistically finds the optimal schedule much faster than the "classical" *branch and bound algorithm*.

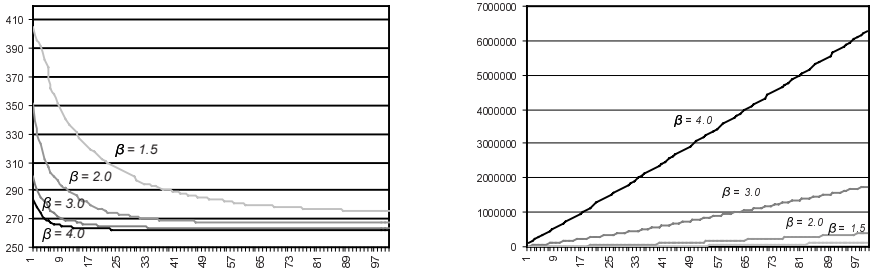


Fig. 3. This figure shows a comparison between different branching factors. Axis x shows the number of iterations. In the left figure the best-achieved performance measure is shown. The diagram on the right-hand side shows the number of required steps. The algorithm with the lightest color had 1.5 as branching factor, the darkest one had 4.0. The other two had 2.0 and 3.0. Every algorithm had 0.1 as learning rate and 0.4 as Boltzmann temperature. We have generated these figures by averaging results of a hundred runs

Some experimental results, too, were presented in the paper. Future research direction could be to improving the system by localizing the temperate of the system in a way that every agent should be to control its own temperature. We also plan to schedule transportation and storage resources. The presented method will be tuned with different function approximators, as well. The approach to the reinforcement-learning problem is slightly different from the classical one, because in the system an agent can take more than one action at once (it can send ants to more than one direction).It is supposed that this feature does not affect the learning abilities of temporal difference learning, however, this may need further investigations.

Acknowledgements. This work was partially supported by the National Research Foundation, Hungary, Grant No. T034632, No. T043547 and the 5th Framework GROWTH Project MPA (G1RD-CT2000-00298).

References

1. Baker, A. D.: A Survey of Factory Control Algorithms That Can Be Implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling, and Pull. Journal of Manufacturing Systems, Vol. 17, No. 4, 297-320, (1998).
2. Baker, J. R., and McMahon, G. B.: Scheduling the General Job-Shop, Management Science, May, 31(5), 594–498 (1985).
3. Davis, L.: Job Shop Scheduling with Genetic Algorithms. Int'l Conf. on Genetic Algorithms and Their Application, Pittsburgh, PA, 136–140 (1985).
4. Della Croce, F., Menga, G., Tadei, R., Cavalotto, M., and Petri, L.: Cellular Control of Manufacturing Systems, Eurapean Journal of Operational Research, Vol. 69, 489–509 (1993).
5. Duffie, N. A., Piper, R. S., Humphrey, B. J., Hartwick, J. P.: Hierarchical and Non-Hierarchical Manufacturing Cell Control with Dynamic Part-Oriented Scheduling. Proc. of the 14th North American Mfg. Research Conf., Minneapolis, 504–507 (1986).
6. Falkenauer E., Bouffouix S.: A Genetic Algorithm for Job Shop. IEEE Int'l Conf. on Robotics and Automation, Sacramento, CA, Apr. 9-11, 824–829 (1991).
7. Hatvany, J.: Intelligence and Cooperation in Heterarchic Manufacturing Systems. Robotics & Computer-Integrated Manufacturing, Vol. 2, No. 2, 101–104 (1985).
8. Haykin, S.: Neural Networks, A Comprehensive Foundation. 2nd Edition, Prentice Hall (1999).
9. Jain, A. S., Meeran, S.: Job-Shop Scheduling Using Neural Networks. International Journal of Production Research, 36(5), 1249–1272 (1998).
10. Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon C.: Optimisation by Simulation Annealing: An Experimental Evaluation. Part II. (Graph Coloring and Number Partitioning), Operations Research, Vol. 39, 378–406 (1991).
11. Kádár, B., Monostori, L.: Approaches to Increase the Performance of Agent-Based Production Systems. Engineering of Intelligent Systems, Lecture Notes in AI 2070, IEA/AIE-01, 14th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, June 4 - 7, 2001, Budapest, Hungary, Springer, 612–621 (2001).
12. Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P.: Optimisation by Simulated Annealing. Science, Number 4598, 671–681 (1983).
13. Lageweg, B. J., Lenstra, J. K., and Rinnooy Kan, A. H. G.: Job-Shop Scheduling by Implicit Enumeration, Management Science, December, 24(4), 441–450 (1977).
14. Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B.: Sequencing and Scheduling: Algorithms and Complexity. In: Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory, S. C. Graves et al., editors, Elsevier, 445–522 (1993).
15. Manne, A. S.: On Job-Shop Scheduling Problem, Operations Research, Vol. 8. 219–223 (1960).
16. Monostori, L., Márkus, A., Van Brussel, H., Westkämper, E.: Machine Learning Approaches to Manufacturing, CIRP Annals, Vol. 45, No. 2, 675–712 (1996).
17. Sutton, R. S., Barto, A. G.: Reinforcement Learning. The MIT Press (1998).
18. Ueda, K.; Márkus, A.; Monostori, L.; Kals, H.J.J.; Arai, T.: Emergent synthesis methodologies for manufacturing, Annals of the CIRP, Vol. 50, No. 2, 535–551, (2001).
19. Valckenaers, P., Van Brussel, H., Kollingbaum, M., and Bochmann O.: Multi-Agent Coordination and Control Using Stigmergy Applied to Manufacturing Control. European Agent Systems Summer School, Prague, 317–334 (2001).
20. Vámos, T.: Co-operative Systems - An Evolutionary Perspective. IEEE Continuous Systems Magazine, Vol. 3, No. 2, 9–14 (1983).

21. Van Brussel, H., Jo Wyns, Valckenaers, P., Bongaerts, L., Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers in Industry, Vol. 37, 255–274 (1998).
22. Williamson, D. P., Hall, L. A., Hoogeveen, J. A., Hurkens, C. A. J., Lenstra, J. K., Sevastjanov, S. V., and Shmoys, D. B.: Short Shop Schedules. Operations Research, 45(2): 288–294, March-April (1997).
23. Zhang, W., Dietterich, T. G.: A Reinforcement Learning Approach to Job-Shop Scheduling. In: Proceedings of Fourteens International Joint Conference on Artificial Intelligence, 1114–1120 (1995).

Agent Architecture for Dynamic Job Routing in Holonic Environment Based on the Theory of Constraints

Leonid B. Sheremetov[1,2], Jorge Martínez[1,3], and Juan Guerra[1,3]

[1] Mexican Petroleum Institute, Mexico
[2] St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS), Russia
[3] Center for Computing Research of the National Technical University, (CIC-IPN), Mexico
sher@imp.mx, {george,jguerrav}@correo.cic.ipn.mx

Abstract. Holonic Manufacturing Systems have emerged over the last decade as strategy for manufacturing control system design to cope with rapid changes in manufacturing environment. Resource and component agents integrated in a holarchy are proposed in this paper to dynamically perform co-operative job routing using a distributed algorithm based on the theory of constraints. Members of the holarchy negotiate and compromise on the optimal production flow in order to meet commitments made to each other. Being performed in a distributed manner, the architecture can increase the agility and responsiveness of an integrated system. This flexible structure has been implemented in an open agent environment using JADE agent platform. The performance based results of the simulation experiments are presented and discussed in the paper.

1 Introduction

The purpose of this work is to develop and study an agent based architecture for reconfigurable manufacturing control which can adapt to continuous changes in the manufacturing environment. The research methodology of this study is inspired in our previous work on agent-based configuring of cooperative supply chains (CSC) [2] and is mainly derived from the theory of constraints (TOC) [8], agent-based computing [10] and holonic manufacturing systems (HMS) [4, 7]. The goal of the HMS approach is to develop an architecture for highly decentralised manufacturing systems, built from a modular mix of standardised, autonomous, co-operative and intelligent components, in order to cope with rapidly changing environments. That is why agent architectures is the natural way to implement intelligent information infrastructure for HMS.

The paper deals with the problems of job routing in holonic environment on the shop-floor level. Recently, the use of agents for scheduling and control of manufacturing systems has attracted attention of many research groups under different perspectives both for traditional [5, 11, 14, 15] and holonic systems [3, 9, 12, 13]. This paper focuses more on computational issues of multi-agent implementation of holonic agents with plug-and-play capability to enhance the control architecture. A generic agent-based architecture of the fabrication/assembly holarchy integrating resource and component holons is discussed. An agent is associated with each

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 124–133, 2003.

instance of a holon. In this paper, a control system is viewed as an arrangement of decision-making and decision execution agents, pursuing their goals. These agents operate by using their knowledge and perceived information about the manufacturing environment to reason about what actions to take in order to satisfy local and global objectives. To cope with the variety of interactions in dynamic organizational context, an algorithm that enables organizational groupings to be formed incorporating mechanisms to ensure groupings act together in a coherent fashion is described in this paper. Finally, the proposed model is applied for the manufacturing control of the job shop composed of two holarchies in distributed heterogeneous multiagent environment using JADE agent platform. The case study deals with production of hypothetic products.

2 Multi-agent Framework for Holonic Manufacturing

The conventional hierarchical structure of FMS is composed of machine, cell, and factory levels with the corresponding control functions associated with each level ranging from real-time control and operation to planning and scheduling functions at the shop-floor level. This architecture encourages each level to become more centralised and horizontally integrated. Basically, this structure is not compatible with the ideal holonic infrastructure.

While working on the CSC configuring, we have defined a generic pattern which is proposed to be applied as a basic holarchy pattern of the fabrication/assembly HMS (Fig. 1) [2]. In this figure, holarchy fragments separated by the dashed lines correspond to the tiers of the CSC as follows: raw materials, manufacturing, assembly. As in the case of a CSC, these holarchies are dynamic virtual groupings of holons joint together by the similarity of technological processes with the goal to produce a final or intermediate products of AB type. Different types of composition of these holarchies can be defined within a HMS.

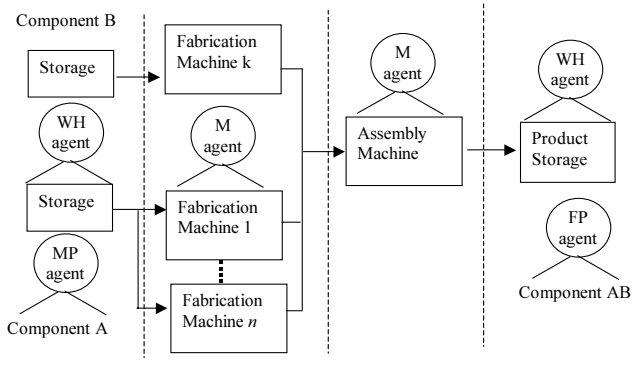


Fig. 1. Structure of a generic holarchy (for the space limits only some agents are shown)

Holons are identified according to the fundamental elements of a physical plant where agents implement the logical part of the holon. As shown in [3], the physical objects

of a manufacturing plant usually can be categorised into two general groups in terms of their properties. One is the *resource,* which performs the manufacturing operations and the other is the *component,* which accepts the manufacturing treatments. Manufacturing control deals with creating routs for components according to their process plan and assigning machines to fulfill operations over components. It is supposed that there is no central control unit that could influence upon a choice of other units. In this case, cooperation rules for the holarchy from the fig. 1 are defined by the technological process as shown in the following sections.

2.1 Resource Agents

There is a set of n agents, $M = \{M_1,...,M_n\}$. This set embraces the machine agents that are present in the system. There is another set of q elements $OP = \{O_1,...,O_q\}$ comprising all the different operations that can be executed by a machine. Each operation O_i is defined by r features contained in a vector of real non-negative values $V_i = \langle v^i_1,...,v^i_r \rangle$. Each v^i_j represent operation parameters such as average time and cost. There is a relation $OM \subseteq M \times OP$ which implies that, the same operation can be offered by one or more machines. On the other hand, the same machine can be designed to perform one or more operations (in a serial rather than parallel way). An additional assumption implies that each V_i vector is linked only to the operation. That is, it does not make any difference which machine performs an operation, the features will not vary from one machine to another.

Machines can fail, this implies that some time must be invested in order to fix the machine and put it on-line again. For simplicity, it can be considered that both failure frequency and repair time are obtained according to a normal distribution and that the same values hold for all machines. The values are: F_{AVG}, which represents the average time a machine works before a failure occurs, F_{SD} is the standard deviation, T_{AVG} and T_{SD} stand for the average time and standard deviation that it takes for a machine to get fixed. Two functions are defined:

$DF : F_{SD} \times F_{AVG} \rightarrow \Re^+$ and $DT : T_{SD} \times T_{AVG} \rightarrow \Re^+$

There is also a set of m agents, $S = \{S_1,...,S_m\}$ denoting raw material storages.

2.2 Component Agents

Two types of component agents have been identified, raw materials (released from a storage) and compound ones (spawn by another component). After the fusion of two components, the integrated agents leave the system and a new agent is generated to continue the process. Component agents work in pairs, hence, one of them is responsible both for retrieving the success or failure state from the other component involved in a composition, and of creating the compound agent.

Formally, there is a set of p agents, $MP = \{mp_1,...,mp_p\}$, where each mp_i represents a type of raw material agent. A function $P: MP \rightarrow S$ returns which storage supplies what raw material. There is a restriction that each one of these must be provided by one and only one storage. There is a special storage agent called Final Product Storage (FPS). Its functionality is focused on a successful reception of final products as well as suggesting upon request, the most urgent items to be produced. We suppose

that this is the only agent with knowledge on product demand. This storage holds q elements set FP = $\{fp_1,...,fp_q\}$ where each fp_i represents a product that this system is able to generate. Products are organized according to a partial order $\preceq$ such that $fp_1 \preceq fp_2 \preceq ... \preceq fp_{q-1} \preceq fp_q$. This expresses the different priorities for each product manufacturing.

If needed, machine and storage agents can be organized according to a partial order $\otimes$ such that $M_1 \otimes M_2 \otimes ... \otimes M_n$. This expresses the way the group of machines and storage feed and are fed with components; besides, it is also possible that machines forming a group can work in parallel. The following constraints are applied: $\neg\exists$ M_i such that $M_i \otimes mp_j$ and $\forall M_i, mp_j$, $M_1 \otimes FPS$, $mp_j \otimes FPS$, i.e., there are no machines feeding raw material storages and the final product storage is always located at the end of the holon template. According to the technological processes, a subset from $\wp(MP)$ holds the possible combinations of intermediate products.

2.3 Technological Processes Composition

For each element from the FP there is a function defining a technological process TP: $FP \rightarrow \wp(\wp(MP) \times OP)$ such that $TP(fp)=\{(\{mp_1,...,mp_p\},O_1),..., (\{mp_1,...,mp_p\},O_q)\}$. Also, there is a partial order $\oplus$ expressing the way components must be processed or assembled, from raw material to the final product. Technological processes are consistent with templates configuration of machines and groups of machines. Thus, there is a correspondence between partial orders $\otimes$ y $\oplus$ such that $\forall(\{mp_1,...,mp_p\},O_i) \oplus (\{mp_1,...,mp_p\},O_j)$ it holds that $M_k \otimes M_l$ where (O_i, M_k), $(O_j, M_l) \in OM$.

An additional set of p agents, $P = \{C_1,...,C_p\}$, stands for the utility agents. The technological process information is generated by an external utility agent. Another two types of utility agents are agents of inference machines for deductive reasoning and interface agents for results visualization. An inference machine is not necessary related to production line issues; however, it was preferred to provide such mechanism as an external service rather than embedding this capability in the agents.

3 Co-operative Control Algorithm

The negotiation approach developed in this article aims in finding the best manufacturing treatment by resource agents for the component holons during their lifecycle. This approach is based on the TOC [8], an approach introduced with the aim of refocusing the way a company is conceived. So far, only related to manufacturing and production lines concepts from the TOC are adapted to the MAS approach. Special attention is given to the identification of *bottle-neck resources* comprising those resources whose capacity is at most the same as the demand. Bottle-necks indicate the rhythm for feeding the system with raw materials. The important issue is balancing product flow over the production line with the market demand. The fact of keeping machines working all the day is not a sign of efficiency. It only increases inventories of intermediate products.

The negotiation algorithm is composed of different algorithms corresponding to the type and role of the executing agent. Activities performed by the raw material agent differ somehow from tasks assigned to those agents that enter in the middle of the system operation. Since component agents work in groups, two type of roles have been defined: initiator and responder. The corresponding algorithms for these roles follow.

3.1 Algorithm for the Component Agent (Raw Material Initiator)

This algorithm starts in the time instant *t* and is repeated each ρ_{bn} time units.

1. A raw material initiator is released.
2. The component requests the final product storage for the next product that should be manufactured. The answer depends on the priorities and current stock.
3. The component requests the technological process required by the selected product. This is a list of operations to be executed over the component that will be stored in L_{TP}, excluding the first one on which it will be joined with some other component.
4. While the size of $L_{TP} > 1$ do:
   a) Take the next operation from the list.
   b) Negotiate with the machines offering such operation.
   c) Get processed: (i) if the result is successful, continue; (ii) else, notify the storage that a new agent can be scheduled in order to compensate the failure. This new agent will be created in *t'* and the next raw material initiator gets scheduled to $t' + \rho_{bn}$ where *t'* is the current instant of time.
   d) If there is only one element in L_{TP}, do:
      i. Request the creation of a partner component.
      ii. Take the next operation from the list.
      iii. Negotiate with the machines offering such operation.
      iv. Get processed.
   e) There are four possible outcomes.
      i. Both components where successfully processed. A compound agent is created. This agent and its partner leave the holarchy.
      ii. Both failed. The storage is notified to release a new agent. Both failed agents leave the system.
      iii. This component was successfully processed but the other did not. The agent requests a new partner and gets standby status waiting for the spare part. If it fails as well, the request is repeated; otherwise, proceeds as in c.i.
      iv. The other component was successfully processed but this one did not. The agent notifies the storage and leaves the system. The other agent assumes the role of initiator for the new spare part.

3.2 Algorithm for Partner Component Agents (Component Responder)

Agents of this type are released with the information on which product they are participating in. Thus, the sequence of steps is:

1. The component requests the technological process required by the selected product excluding the first opertion in which it will be joined with the initiator.
2. While the size of $L_{TP} > 1$ do:
   a. Take the next operation in the list.
   b. Negotiate with the machines offering such operation.
   c. Get processed: if the result is successful, continue, else, notify the initiator.
3. If there is only one element in L_{TP}, do:
   a. Take the next operation from the list.
   b. Negotiate with the machines offering such operation.
   c. Get processed.
4. There are two possible outcomes:
   a. Both components were successfully or unsuccessfully processed. Responder component leaves the system since its partner is in charge of dealing with these situations.
   b. The initiator component failed, but this one did not. Hence, it takes the initiator role and gets idle status while waiting for the new spare part to arrive.
      i. If the spare part is successfully processed, a compound agent is created. This component and its partner leave the system.
      ii. If the spare part failed, this component requests the respective storage for a new partner.

3.3 Compound Components and Spare Parts Algorithms

As the name implies, these components are the result of components composition. Spare part agents are released with knowledge on what product they are participating in. Also, they receive information about the location of the component already waiting for them. The algorithm is basically the same as the algorithm for raw material initiators. Two exceptions are made: there is no need to ask for what product it will participate in and, the execution is out of the cycle that embraces the first algorithm.

4 Open Agent Architecture

The above described model has been implemented in an open environment using JADE (Java Agent DEvelopment Framework) as the agent platform [1]. Basic agents include instances of all the agent types mentioned in Section 2. Also, two utility agents were implemented. The first one is an output interface for results visualization, which consists of a wrapper agent (WA) to the OpenOffice SCalc worksheet which allows real-time statistical and graphical analysis during a simulation. The second one is an inferences agent (used by component agents) in charge of feeding information about technological processes. It consists of a WA to the SWI-Prolog interpreter under which, a few rules concerning the OP set have been specified for this case of study. The implementation focused strongly on the fact that, agents interaction is based on services request and providing. Fipa-Request, Fipa-Query and Fipa-Contract-Net protocols are intensively used during agents interaction [6]. The latter one supports negotiations between machines and components.

4.1 Implementation of the Negotiation Algorithm

The implementation was aimed to simulate the developed algorithms. The Scheduler role is assigned to the bottle-neck machine. Also, this machine is in charge of starting the simulation by requesting the release of the first and subsequent raw material agents. This task is repeated according to the speed of the bottle-neck machine and is registered on an internal counter representing the system's time. Conversations between machine and storage agents are engaged via the Fipa-Request protocol.

After the component agent is aware of what type of product must be produced, it sends an ACL message to the Prolog WA. The content includes the type of component and intended product type. The answer comprises two parts: a list of operations the component needs to go through and the type of component required to generate a compound item. From this point, the component engages in negotiations with those machines capable of performing the next operation. Each machine sends the time on which it will be available for processing. Agent's choice is based on the earlier moment from the received proposals. Fipa-Contract-Net protocol is used during negotiation. The component agent estimates the time it may take him to reach the assembly machine, the one where it will meet with another component. A Fipa-request protocol is used for requesting a storage to release the required partner (responder).

The responder agent repeats the process of querying the technological process to be accomplished. Nevertheless, this agent already knows who is waiting for it and at what time. By negotiating with the respective machines, the agent proceeds to synchronize its arrival time with the initiator agent. Once both agents have completed successfully their list of operations, the initiator spawns a new compound agent representing the fusion of the two participants. As a consequence, these latter abandon the system.

Machines are supposed to fail sometimes during the day. The affected component is discarded and a new spare part is requested as soon as possible. The respective partner re-synchronizes the arrival to an assembly machine and gets on idle at the output buffer of the machine closest to the assembler.

4.2 Hardware Infrastructure

The project is entirely based on the JADE 2.6 platform and programmed in JAVA. The system was codified and tested over the following heterogeneous hardware platform (fig. 2): a Sun SPARCStation 5 holding the JADE platform GUI, a Sun Ultra 10 holding the Prolog wrapper, a Dell Optiplex PC (Pentium II) holding a holarchy, a Dell PowerEdge Server (Pentium III) holding a second holarchy. Finally, two types of computers were used for the results visualization agent (running OpenOffice): an IBM PC (Pentium II) and a Toshiba portable computer (Celeron).

4.3 Experimental Results

Experiments were focused on two main objectives: first, to study the system's behavior under the proposed production model, and second, to evaluate the implementation performance over the selected platform (JADE). The following configurations were considered. Holarchy 1 is composed of a set of holonic agents

comprising two raw material storages (S1 and S2), five machines and three components. S1 feeds type A component agents to three machine agents: M1 and M2 performing O1 and O2 fabrication operations respectively while M3 is able to perform the same operations as M1 and M2. S2 feeds type B components to M4 (the holarchy's bottle-neck), which is enabled with two types of alternative operations, O3 and O4. Finally, M5 assembles processed components from the above machines. Holarchy 2 is a set of agents comprising one storage (S3) that feeds M8 (the bottle-neck) able to fulfill two operations (O7 and O8). M6 and M7 receive and process AB components from the first holarchy. They are parallel machines enabled to do the same operation, O6. Finally, M9 assembles AB with C components and delivers the result to a FPS.

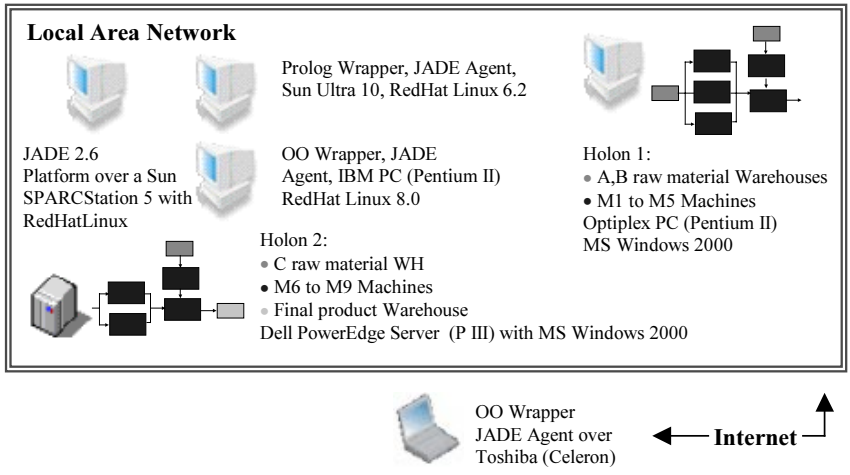


Fig. 2. Implementation hardware platform

Figure 3.a shows operation, idle and repair time for the eight machines obtained during simulation. M1, M2 and M3 show a high idle time. The reason is that they share common operations exceeding the system's demand. A similar effect is found for the pair M6-M7. However, the most important machine to consider is the bottle-neck M8. It can be noticed that M8's idle time is also high, an unacceptable performance according to the TOC. M8 must stand as the machine that never stops, unless a failure breaks the normal functioning. A high sensibility of M8 to failures of the preceding machines processing B and AB components has been detected. In other words, M8 reflects wasted time from failures in machines that operated over the B component. However, this is also related to the chosen technological processes for the experiments. The experiments showed remarkable low levels of inventory during the simulation.

The second type of experiments was divided into three phases: all the agents executed in the same computer (PC PowerEdge Server), a distributed approach over a LAN (as shown in Figure 2), and a second distributed approach with the OpenOffice WA connected through Internet from a distant country. The experiments included 6, 12 and 24 hours production program simulations for each modality. Figure 3.b shows the relationship between the real production time and the simulation time. From the beginning, it was noticed that the OpenOffice WA required intensive communication

with the rest of the group (more than 9,000 messages for a day program). Due to this reason, there was also a distinction between simulations with this agent present or absent.

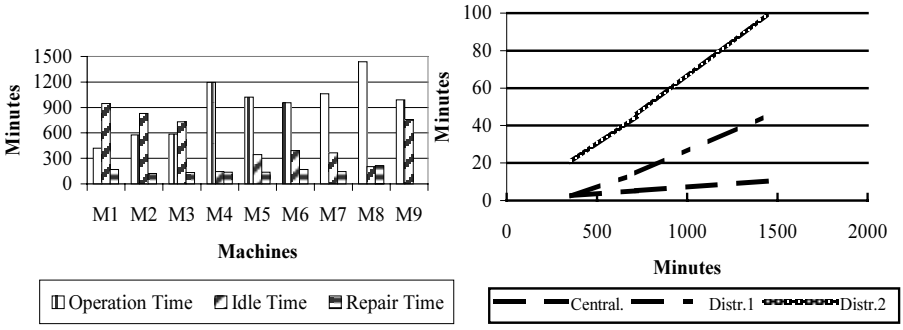


Fig. 3. Experimental results: a) Machine operation, b) Platform performance.

The results show that all the simulations where the WA is absent, require approximately the same time to finish. Yet, it comes up that when the WA is considered, the time to process all the messages grows significantly. Network traffic and distance among computer nodes are not considered. Finally, in the simulations without graphic WA, it was found that the LAN experiment required less time to finish. This is a clear effect of workload distribution. Nevertheless, it was also found that in the Internet experiment, the time grew again with respect to the other experiments.

5 Conclusions and Future Work

With successful cases in the field of manufacturing, the theory of constraints was considered an adequate approach to be combined with HMS/MAS perspective. According to this theory, a company must be optimized as a whole, it is no longer a set of isolated components under collaboration. A global interaction plan is required under which agents will collaborate (or compete) without forcing them to engage relationships reducing their autonomy. This plan is negotiated by the resource and component agents within each holarchy using a distributed algorithm.

Proposed architecture permits to create reconfigurable and scalable control system when the manufacturing equipment or the production process is changed. Each time a new product fp_i is introduced in the holarchy, a resource holon is employed to perform a new operation to achieve the plan. A new holarchy can be also composed dynamically and temporarily if required by the fp_i production process. The change implemented by the holonic controller is to plug the software component of the corresponding resource holon as well as to plug the software component of a new product fp_i into the holarchy controller. The manufacturing plan is immediately changed without causing any side effects to the other parts of the system.

Currently, the system does not allow component agents to change their objectives on the fly. A new component as well as its partner, are engaged with the same type of product. However, the products are dynamically scheduled, as raw material initiators enter the system. Additional analysis on how to locate a raw material agent triggering

the composition of a final product is required. In the ongoing work, different control strategies are studied. Current experiments are also focused on further investigation of the interoperability issues with software modules that provide additional services for agents. Finally, we will try to apply the methodology integrating this MAS into the enterprises framework.

Acknowledgements. Partial support for this research work has been provided by the CONACyT, Mexico within the project 31851-A and by the IMP, within the project D.00006.

References

1. Bellifemine F., Poggi A. & Rimassi G., JADE: A FIPA-Compliant agent framework, Proc. Practical Applications of Intelligent Agents and Multi-Agents, April (1999), 97–108
2. Chandra, C., Smirnov, A., and Sheremetov, L., Agent-based Infrastructure of Supply Chain Network Management, In Camarinha-Matos, L.M., Afsarmanesh, H. and Rabelo, R. (Eds.) E-Business and Virtual Enterprises Managing Business-to-Business Cooperation. Kluwer Academic Publishers, (2000), 221–232.
3. Chirn, J.L. and McFarlene, D. C., A holonic component – based approach to reconfigurable manufacturing control architecture, In Proc of HolonMas00, London, (2000).
4. Christensen, J., Holonic Manufacturing Systems – Initial Architecture and Standards Directions, 1st Euro. Conf. On HMS, HMS, Hanover, Germany, 1994.
5. Denkena B, Tonshoff H.K., Zwick M., Woelk P-O. Process Planning and Scheduling with Multiagent Systems. In V. Marík, L. Camarinha-Matos, H. Afsarmanesh (Eds.) Knowledge and Technology in Product and Services. Selected papers of the IFIP, IEEE International Conference BASYS'02, Kluwer Academic Publishers. (2002), 339–348.
6. FIPA Iterated Contract Net Interaction Protocol Specification, Foundation for Intelligent Physical Agents, URL: http://www.fipa.org.
7. Fletcher, M.; Garcia-Herreros, E.; Christensen, J.H.; Deen, S.M.; Mittmann, R.; An open architecture for holonic cooperation and autonomy. Proceedings of the 11th International Workshop on Database and Expert Systems Applications, (2000), 224–230.
8. Goldratt E. M. The Goal. North River Press Publishing Corporation, Great Barrington, Massachusetts, (1992)
9. Heragu, S. S., Graves, R. J., Byung-In Kim, St Onge, A. Intelligent agent based framework for manufacturing systems control. IEEE Transactions on Systems, Man and Cybernetics, 32(5): 560–573, (2002).
10. Jennings, NR. On agent-based software engineering. Artificial Intelligence, 117: 277–296, (2000).
11. Lin, G.Y. and Solberg, J.J., An agent-based routing manufacturing control simulation system, in Proc of Winter Simulation Conference, (1994), 970–977.
12. Maturana, F., Shen, W., and Norrie, D.H., MetaMorph: an adaptive agent-based architecture for intelligent manufacturing, Int. J. Production Research, 37(10), 2159–2173. (1999)
13. Misbah Deen S. Cooperating Agents for Holonic Manufacturing V. Mařík et al. (eds.); MASA 2001, LNAI 2322, Springer-Verlag Berlin Heidelberg (2002), 119–133.
14. Reveliotis, S. A., Production Planning and Control in Flexibly Automated Manufacturing Systems: Current Status and Future Requirements, IEEE ICRA, (1999), 1442–1449.
15. Usher, J.M., Negotiation-based in job shops via collaborative agents, Third International ICSC Congress on World Manufacturing WMC'2001, Rochester, N.Y., Sept. 24–27, (2001).

A Heterogeneous Multi-agent Modelling for Distributed Simulation of Supply Chains

Olivier Labarthe[1, 2], Erwan Tranvouez[1], Alain Ferrarini[1], Bernard Espinasse[1], and Benoit Montreuil[2]

[1] LSIS UMR-CNRS 6168, Polytech'Marseille, Université Aix-Marseille III
Domaine Scientifique Saint Jérôme, 13397, Marseille Cedex 20, France
{olivier.labarthe, erwan.tranvouez, alain.ferrarini,
bernard.espinasse}@lsis.org
[2] CENTOR, Network Organization Technology Research Center, Université Laval, Ste-Foy,
Québec, G1K 7P4, Canada
Benoit.montreuil@centor.ulaval.ca

Abstract. This paper presents a heterogeneous (cognitive/reactive) agent based approach to model supply chains. The proposed model based on an actors' representation introduce the behavioural studies of active entities constituting the logistics organization. Supply chains member's behaviours are split up into two categories: deliberative and operational. The design and exploitation of distributed simulation model with multi-agents systems permits to support the representation of entities realizing decision-making and operational activities. To facilitate the design of such models, a dedicated agent model is proposed for each category of behaviour: the Decision Agent and Simulation Agent.

1 Introduction

Simulation is an abstracted representation of a real system and involves the elaboration of models which must reproduce the behaviour of the system following a set of hypothesis used to define different scenarios. Multi-agent simulation focuses on distributed behavioural description and study in a dynamic context. Multi-agents systems (MAS) relies upon autonomous entities simulating the complex behaviour of components whose interaction dynamics explain the real system global behaviour. Multi-agent simulation models consist in identifying such entities (agents), their behaviours, their interactions among themselves or with the environment in which they are situated. Supply chains are composed of companies modelled as actors realizing activities in a logistics network. Actors' behaviours are split into two categories: deliberative and operational. Deliberative behaviours relate to decision-making processes achieved by some actors in the system. Operational behaviours consider how such decisions are realized. Our multi-agent modelling approach allows dissociating representation between decision-making level and operational activities level. This approach facilitates the behavioural specification and the *agentifying* phase in a heterogeneous context.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 134–145, 2003.

Our work proposes a design approach for simulation models by heterogeneous MAS. The second section presents general principles underlying supply chains and related works on agent based supply chain management. The third section introduces the model of the Actor Agent enabling the representation of supply chain entities. The fourth section illustrates this modelling applied to the field of supply chains. Next section shows how agents' behaviours are specified as well as the interactions between deliberative and reactive agents. Finally we conclude and give some perspectives of our research work.

2 Supply Chain: From Network of Actors to Agent Society

Supply chains architecture constitutes an economical advantage for companies and represents economic and social systems. This policy allows apprehending the whole product transformation process from the first supplier to the final consumer. Supply chains are composed of a set of sub-systems organized to reach their own purposes and/or objectives made possible by information exchange. The information processing and analysis allows the organization to adapt its behaviour to the environmental dynamics. Information induces then a process of permanent adaptation of the organization to modifications of environment characteristics. Through concepts such as autonomy and cooperation, multi-agent systems have demonstrated their ability to provide a modelling and simulation framework for industrial systems and in particular supply chain [9]. However, most works such as DASCh [10], MASCOT [12] and the Supply Chain Demonstrator [13] deal mainly with homogeneous agent societies composed of either deliberative (decision-making ability) or reactive (simplified predefined behaviours) agents.

Our work focuses on supply chain control based on performance measurement. This aim leads us to consider how decisions are made locally and executed as well as their impact on the local and global performances. Thus our modelling approach identifies a set of interacting constituents (micro analytical approach) with different coordination, cooperation and negotiation abilities [3]. We define such abilities according to three decisional levels (strategic, tactic and operational). This is done in order to fulfil own actors' strategies as well as the emergent global strategies of the chain. These different action-able and social constituents are denoted actors: the supply chain is therefore modelled as a network of actors prior to its operationalization into a heterogeneous agent society.

3 Modelling Approach by Multi-agent Paradigm

This section introduces the concept of Actor Agent enabling the study and the analysis of complex systems according to an agent-based modelling approach. The micro analytical approach considers a system as a set of interacting entities is called Individual Based Modelling in ecosystems modelling field. This modelling approach allows the representation of the dynamic characteristics of a system. This involves the

specification of the actions, events, behaviours and interactions between the different "individual" entities composing the population system. The term of "individual" is not restricted to an anthropomorphic sense (an individual = an atomic unit).

3.1 The Actor Agent Model

The social characteristics of each individual, i.e. the roles it plays as well as its responsibilities, affect the decision-making processes carried out in a system. Such decisions results in actions executed by actors of the organization. Addressing these two points of view involves the exploitation of events, actions and interactions between entities of the complex system. To this end, we propose to model complex dynamic systems as a set of actors. An actor is defined as a social entity, an individual or a group of individuals (like in [1]). An actor of the system represents either a decision-making capable entity (deliberative activity) or an operative entity (operational activity). In the real system each type of activity is associated to a particular level of granularity and represented in the Actor Model {Decision Centre, Physical Resource}. The Actor Agent relies upon a simplified model of the reality allowing a behavioural study through the couple {decision, action}. The elaboration of the simulation model requires identifying the entities in the real system and activities associated. Same activities are gathered into one or several Decision or Simulation Agent. The Actor Agent is composed by agents (deliberative or reactive) modelling different activities of an actor.

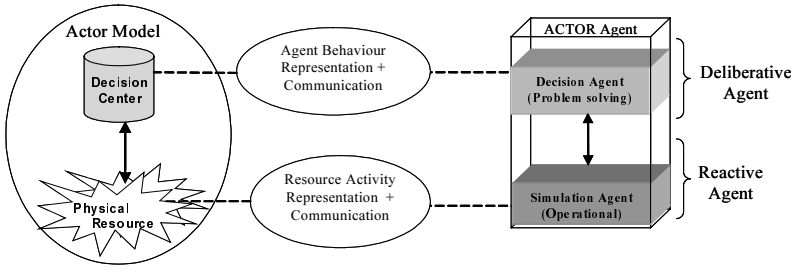


Fig. 1. From Actor Model to Actor Agent

Decision Agent represents the Decision Centre decision-making activities of an actor of the system and transmits decisions to Simulation Agent. Simulation Agent reproduces the behaviour of an operational activity and transmits signals to the Decision Centre to report activities results. Figure 1 illustrates this duality in the Actor Model and shows which behaviour each agent adopts. Thus, we have specialized and adapted the agent's internal architecture to the activities they had to realize. The Decision agent is based on a deliberative agent architecture allowing the representation of knowledge as well as rules using this knowledge and plans specifying complex behaviours [14]. The Simulation Agent relies upon a reactive architecture and more precisely "reflexes with states" [11]. The Actor Agent structure can represent an actor with a single agent either deliberative or reactive depending on the characteristics of the "individual" entities modelled.

3.2 The Actor Agent Model Applied to Supply Chains

The logistic network is composed of a set of actors which have roles and additional responsibilities defined from the competences and activities they are able to realize. When the network responsibility is set up by the pooling of local strategies the structure of the collaborative actors' network emerges. The responsibilities which belong to the actors according to the supply chain level decomposition define new responsibilities for the actors at the companies' level (decision centres). They can coordinate theirs decision-making activities that lead to the fulfilment of local strategic objectives. This method of responsibility decomposition and distribution is in a similar way applicable to intra company level (activity cells). Actors of these cells have the responsibility for realizing the operational activities by use resources. Figure 2 shows the modelling of supply chain actors.

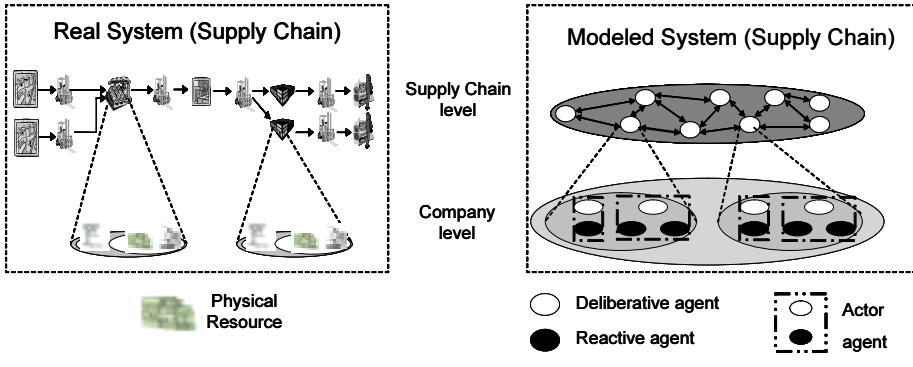


Fig. 2. Decomposition of the Supply Chain

This mode of representation based on an organisational point of view makes it possible to exhibit relationships and interactions between actors of the supply chain according to the responsibilities decomposition and distribution levels. This mode of representation necessitates a complete operational point of view. In the model, the following processes must appear: the items movements, the registering and the state of available resources, the operative activities realized by the actors and the data needed at each decision levels affecting the flow traffic.

4 A Heterogeneous Multi-agent System

MAS consists of a set of autonomous entities called the agents. Every agent possesses one or more roles defining the expertise and behaviour enabling him to act in an autonomous way. The general behaviour of the system emerges from the realization of the individual agents' actions. Our modelling approach is based on heterogeneous MAS as it is composed of two different architecture agents. Agents of a same kind form a society with uniform interactions mode (message or signals). Both societies

interact through the actor link between decisional and operational agents. Actions realized by reactive agents influence decisions emitted by deliberative agents, and conversely. This section describes how these agents societies manage to form a MAS.

4.1 Architecture of Heterogeneous Multi-agent System

Within our modelling framework, an entity of a real system will be represented by an Actor Agent called Decision Agent to realize deliberative processes for problem solving and one or several reactive agents called Simulation Agent to perform operative activities. The elaboration of multi-agent simulation models should be flexible [6] and adapted according to the types of behavioural studies. These models represent individual actions of actors' interactions as well as consequences of these interactions on the environment. This approach allows dissociating the conception of MAS from the application domain.

The model of the physical system represented by a MAS can be decomposed into sub systems. Every system or physical sub-system consists of a set of Actor Agent. The MAS architecture proposed consists of two societies of heterogeneous agent: deliberative and reactive agents. The Actor Agent allows establishing linkages between both agents' societies. Agents of the deliberative agent society can act in an autonomous way to attain their own objectives. They have an explicit representation of the environment and reasoning capacities based on knowledge. Communications are needed by the deliberative agents' society to describe elaborate conversation for complex interaction processes. Communication acts are used to determine the supply chain control and performance (example: negotiation protocol to define new contracts, obtain new delivery dates, …). The agents can play several roles within the MAS. Agents of the reactive society react in answer to stimuli from their environment. They don't have an explicit representation of the environment and their behaviours are predefined by "simple rules". They are not able to reason about their intentions.

Structure of the MAS concerns the description of entities possessing properties and relations among them according to the agent society they belong to. The MAS architecture enables the representation of entities' behaviour of the real system. Our approach relying upon heterogeneous agent society proposes to dissociate decision-making, operational and informative processes within the MAS. This decomposition allows distinguishing the behaviour, the roles and competence adopted by entities.

4.2 Multi-agent System Applied to Supply Chain Modelling

The supply chain model presented here is formalized relying upon the model of the Actor Agent. It allows obtaining a heterogeneous MAS constituted of deliberative and reactive agents. The principle of heterogeneousness is to enable the separation of the representation of decision activities from the operation activities. In the supply chain

model, deliberative agent societies assume the behaviour of the supply chain actors which implement decision processes at strategic, tactic and operation level. Reactive agent societies assume the behaviour of+ the physical resources employed by deliberative agent societies, to realize transformation activities. Next figure presents the organisational model coupled with the supply chain operational model for the behavioural modelling of operation and decision activities in the logistic network.

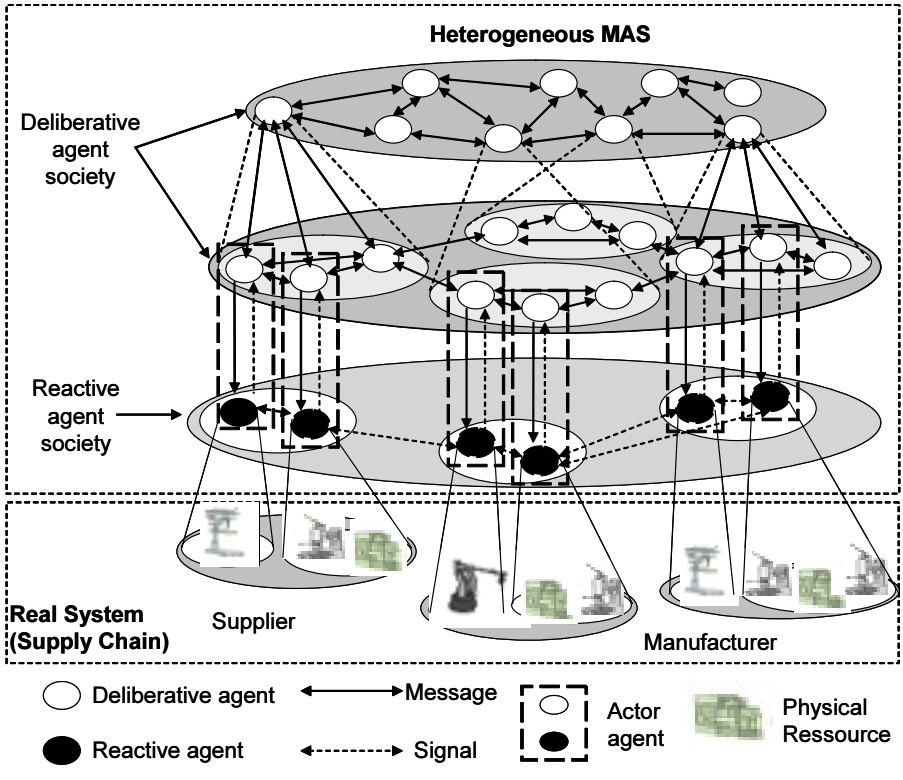


Fig. 3. Heterogeneous MAS Supply Chain modelling

This approach combines the representation of individuals and their behaviour nature. Figure 3 shows how agents gather to form actor as well as the different interaction types between agents (signal or message). This information summarizes the real system organisation according to several levels: the individual, groups (actor or group of actors) and the whole system. Modification of the structure system during time should appear according to interactions established among individuals of the population. This approach of modelling is adapted to simulate organized complex systems of which global functioning emerge from individuals' actions.

5 Agents Specifications and Interactions

5.1 Specification of Decision Agents Behaviours

As expected from a deliberative agent, the Decision Agent have an explicit representation of the environment in which it evolves, take into account its modifications during their deliberative processes and communicate through exchanges of messages. The part of the simulation based on such agents result from the cooperation of a relatively small number of agents, allowing for example, tasks sharing to solve a complex problem [7].

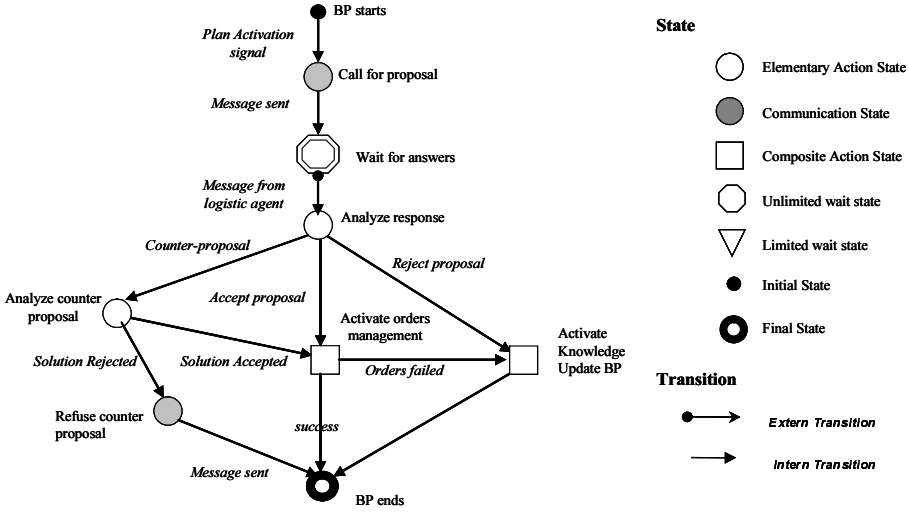


Fig. 4. Notation of ABR formalism and example of Decision Agent behaviour

The deliberative agent model can be related to a Belief Desire Intention model [4] as its behaviours are explained and designed with plans. These plans, denoted Behaviour Plans, are used to specify and execute (after a phase of translation in a target rule-based language) the behaviour of the agents according to two models: individual and social [14]. Within the individual model is represented the autarkic behaviour of the agent (Local Plans), i.e. interactions with the other agents are not required to reach its goals.

The social model involves the social behaviour of an agent (Role Protocols) and thus specifies how an agent interacts with others (message passing) as well as the actions leading and resulting from these interactions. The set of Behaviour plans of an agent describe its ability to react and solve problems whether communicating with other agent or not.

These plans are specified with the ABR Formalism (Agent Behaviour Representation) [14] which consists in a state graph formalism with strongly specialized states and transitions :

- A state represents what the agent is doing i.e. computing or sending a message for example in an action state or waiting for a message in a wait state.
- A transition is fixed depending on the outcome of an action (internal event) or a message arrival (external event) and in turn activates another state.

The different states composing behaviour plans describe the sequence of actions an agent has to perform in a particular context.

The description can be recursive as a state can itself be a behaviour plan. As indicated before, such plans can be rather easily translated in a rule-based language, and thus can be reused to guide the implementation phase of Decision Agents. When alive, the agent will monitor its environment and react accordingly to its modification by instantiating the corresponding plan. Theses plans are also used to state how an agent can reach its goals. This deliberative agent architecture is already defined and operational [14].

5.2 Specification of Simulation Agents Behaviours

As reactive agents, Simulation Agents' behaviours are restricted to simple sequences of actions consisting in reacting to signals and emitting in turn new signals. They do not represent themselves the environment in which they evolve and have a simplified communication device. They do not possess an intelligent individual behaviour even if their global behaviour through their interactions satisfies a purpose [15].

Activities of a Simulation Agent are specified with the UML statechart diagrams formalism [2]. The statechart diagrams describe the state of the operational agents in function of external stimuli and/or its current activity (e.g. *mould for 1 minute*). Stimuli represent interactions with other agents (reactive or deliberative – signal or message) or objects of the environment (a product):

- A state is characterized by a duration time for the action to be realised.
- A transition represents the immediate passage from a state to another one. A transition is activated by a particular event (e.g. a stimulus).

Figure 5 proposes an example on how a production activity can be specified.

5.3 Hybrid Architecture of Actor Agent

The Actor Agent model is composed of two different agent types of different granularity. This difference of granularity implies using two different formalisms for the representation of their specific behaviours. The Actor Agent represents the actor

of the real system as a whole, i.e. encompassing both aspects of its behaviour. This modelling approach helps to easily pass from an Individual Based Modelling to an executable multi-agent modelling. Decision and Simulation Agents reproducing an actor complex behaviour are thus linked by their real world identity. Figure 6 illustrates how the Decision Agents are linked together: the Decision Agent(s) make decisions communicated to the Simulation Agents where they are carried out.

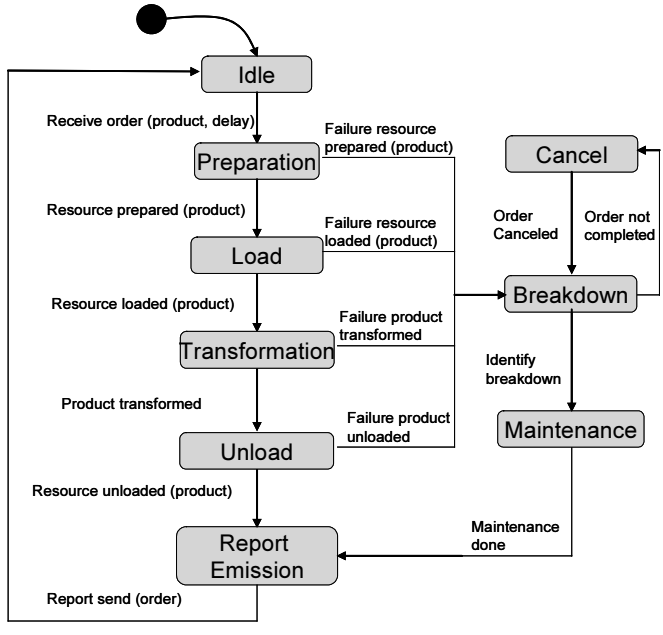


Fig. 5. Example of Simulation Agent behaviour (UML)

The Decision Agent deliberative architecture is based on BDI architecture [14] which integrates the ABR formalism. This integration relies on tools helping the translation of ABR graphs into Jess (Java inference engine). Decision Agents communicate by exchange of ACL FIPA messages. When directed to its Simulation Agents a message corresponds to an information request or command. Simulation Agent architecture is restricted to a module executing simple behaviours specified with UML state diagram formalism.

The second module translates signals and messages received from other agents into events which activate a transition. It monitors the reactive agent activity to redirect some specific events to other agents. Simulation Agent adapts its behaviour in function of the Decision Agent orders and signals receive and can answers to signals perceived by other Simulation Agents.

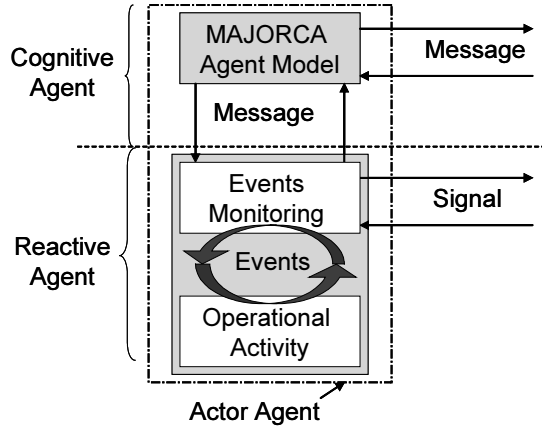


Fig. 6. The Actor Agent model

5.4 Interactions Representation

Simulation is constituted of a set of agents whose states changes depend on interactions with the environment. To coordinate agents' behaviour and manage their organizational structure it is necessary to define languages and protocols of communication among agents. Interactions among deliberative and reactive agents are specified by exchange messages following the ACL-FIPA standard. Agent UML is based on the Unified Modeling Language method used for software engineering with object-oriented languages [8]. AUML replaces the notion of method by the notion of service and the notion of class by the notion of goal. This type of representation allows showing the type of exchanged messages and their contents as well as the process of activities which they activate [5].

Orders are a transfer of information from deliberative agents towards reactive agents. Deliberative agents use messages to communicate among them. Reactive agents emit signals with the deliberative agents who interpret them in the form of information. Canals of transfer are bidirectional. Description in natural language help in the phase of MAS design during to represent the physical model.

6 Conclusion

This paper has presented a multi-agent modelling approach aiming at simplifying the transition phase from Individual Based Modelling to Distributed Simulation architecture. This objective relies upon a unifying concept: the actor. The Actor Agent model distinguishes two agent types to model actors' activities: Decision and Simulation agent. The proposed approach support designers and users during the

modelling phase by providing different level of abstraction for the system behaviour specification.

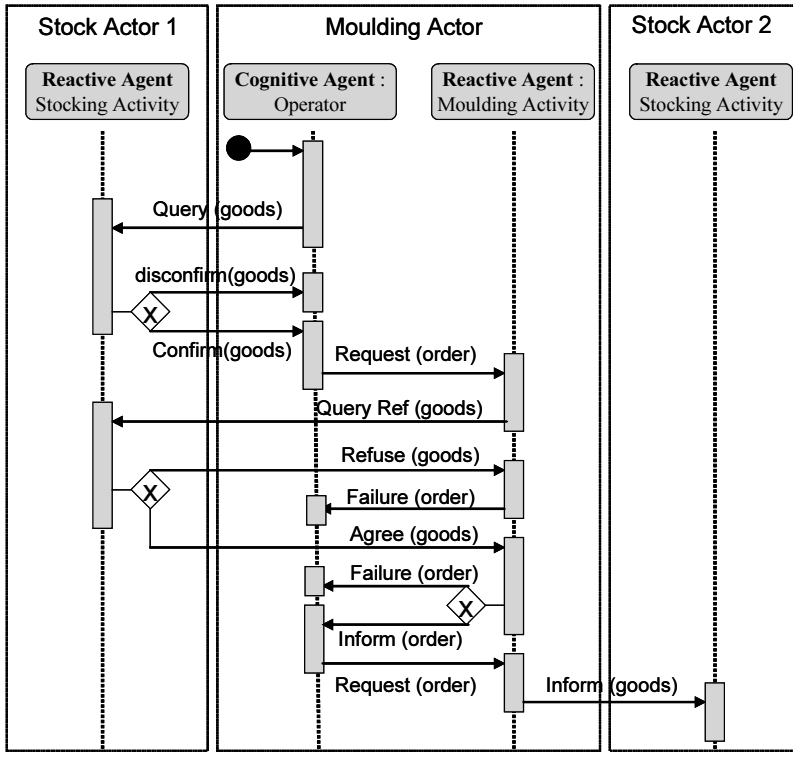


Fig. 7. Sequence diagrams (AUML)

The work presented here constitutes the first stage leading us towards a multi-agents based simulation platform dedicated to the heterogeneous behavioural study of supply chains. Our experience in former agent development projects will support this phase. Agents controlling the simulation process are already identified but not presented for lack of space will be integrated in this heterogeneous architecture.

References

1. Amblard, F., Ferrand, N. : Modélisation multi agents de l'évolution de réseaux sociaux. Modèles et SMA pour la gestion de l'environnement et des territoires, pp 153–168 (1998).
2. Booch, G., Rumbaugh, J., Jacobson, I. : The Unified Modeling Language User Guide. Addison-Wesley, Reading, Massachusetts (1999).
3. Ferrarini, A., Labarthe, O., Espinasse, B. : Modelling and Simulation of Supply Chains with a Multi Agent System. 13th European Simulation Symposium, France (2001).

4. Fisher, K., Müller, J.P., Pischel, M. : Cooperative transportation schedulling: an application domain for DAI. Applied Artificial Intelligence, vol. 10, pp 1–33 (1996).
5. Huget, M.P. : An Application of Agent UML to Supply Chain Management. Proceedings of Agent Oriented Information System (AOIS-02), Italy (2002).
6. Meurisse, T., Vanbergue, D. : Et maintenant à qui le tour? Aperçu de la problématique de conception de simulations multi agents. Laboratoire d'Informatique de Paris 6 (2001).
7. Moulin, B., Chaib-Draa, B. : An Overview of Distributed Artificial Intelligence. Foundations of Distributed Artificial Intelligence, O'Hare and .Jennings, pp 3–55 (1996).
8. Odell, J., Parunak, H., Bauer, B. : Representing agent interaction protocols in UML. AAAI Agents conference, Spain (2000).
9. Parunak, H. : What can Agents do in Industry, and Why? An Overview of Industrially Oriented R&D at CEC. Industrial Technology Institute (1998).
10. Parunak, H., Savit, R., Riolo, R.L. : Agent-Based Modeling vs. Equation-Based Modeling: A case Study and User's Guide. Center for Electronic Commerce (1998).
11. Russell, S., Norvig, P. :Artificial Intelligence: A Modern Approach. Prentice-Hall (1995).
12. Sadeh, N., Hildum, D., Kjenstad, D., Tseng, A. : MASCOT: An Agent-based Architecture for Dynamic Supply Chain and Coordination in the Internet Economy. Production Planning & Control, Vol. 12, n° 3 (2001).
13. Teigen, R., "Information Flow in a Supply Chain Management System", Ph. D. Thesis, University of Toronto (1997).
14. Tranvouez, E., Espinasse, B. : Protocoles de coopération pour le réordonancement d'atelier. Journées Francophone sur l'intelligence Artificielle distribuée et SMA (1999).
15. Wooldridge, M., Jennings, N.R. : Agents Theories, Architectures, and languages: A Survey. Notes in Artificial Intelligence, Wooldridge and Jennings, pp 1–39 (1994).

Integration of Shop Floor Holons with Automated Business Processes

Klaus Glanzer, Thomas Schmidt, Gerald Wippel, and Christoph Dutzler

Vienna University of Technology – ACIN 376
1040 Wien, Austria
{Glanzer, Schmidt, Wippel, Dutzler}@acin.tuwien.ac.at

Abstract. Changing customers' behavior require a dramatic change for future manufacturing control concepts. Emerging technologies for distributed application, networking, and intelligent information processing pave the way to reach economic lot size one manufacturing systems. Holonic Manufacturing System provides an approach aiming at this objective. The approach proposed in this work deals with the enhancement of Holons by Web Services for seamless integration in business processes and facilitated reconfigurable shop floor control.

1 Introduction

Customers' desires of customized products force enterprises to strive economic lot size one production. This ambitious objective requires high flexible production systems as well as tight integration with product development and logistics.

Recent developments in different areas open new potentials. Cheap, approved and network-compatible microprocessors allow equipping each resource on the shop floor with local intelligence for information processing. Internet technologies and Internet approaches deal with, among others, communication, interoperability and security of distributed, networked, and loosely coupled applications. Picking up proper technologies from different fields and merging them yield new chances.

Maximizing the return on invest is the fundamental requirement to production sites or every investment a manager would agree on. From the engineers point of view this requirement could be specialized into three categories: functional level, system level and integration level requirements. Naming some important: functional level requirements comprise real-time control, on-line diagnosis, in-process data access, etc; system level requirements: efficient reusable engineering, dynamic reconfiguration, scalability, extendibility of running processes, minimize system downtime, handling of high volume and high variety production, etc; on integration level: seamless integration with business processes, instant production data access, and quickly changing set up in production. All facts together point towards agile manufacturing.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 146–155, 2003.

2 Holonic Manufacturing System Approach

A holonic manufacturing system (HMS) is a production system having cooperative and autonomous characteristics. These characteristics are a consequence of the system's building components called holons. Holon is a term devised in [1] to denote a system unit that is simultaneously whole (Greek: holos) and part (Greek suffix: on). This concept implies a recursive structure of holonic systems, which should mean a holon can be part of another holon or a holon can be a compound structure of holons. Holons exhibit two main behavioral tendencies they are self-assertive and integrative.

Now bringing these abstract concepts to manufacturing control. A holon is an autonomous and cooperative building block of a manufacturing system for transforming, transporting, storing and/or validating information, and/or physical objects [2, 3]. We will here emphasize the linkage of holons to physical units for production and material handling. The typical characteristics of holons are autonomy, the ability of cooperation and openness. We can imagine for example in manufacturing plants typical candidates for holons are (mobile) robots, transport carriers, assembly stations, machining centers and so forth.

Holons characteristics directly address the ubiquitous problem of system reconfiguration in manufacturing. Fast reconfiguration in an economic way is a key to be competitive and will be a mandatory key to stay competitive in future markets. Fast reconfiguration obviously is a relative expression. It could be quantified as the ratio of the sum of product design/change time, process redesign time, rescheduling time, machinery set up time respectively set up change time, and information distribution time for a customized product to the standard process time.

Figure 1 [4] shows the principal architecture of shop floor holons. They are structured in three layers. The bottom layer is the physical layer comprising the shop floor resource the mechanics, sensors and actuators. Sensor signals and actuator commands are processed in the second layer the real-time control. On the top the "intelligent" information processing unit, the intelligent software agent, is located. Agent responsibilities are to pursue their committed goals through cooperation with the other system units by communication, collaboration, negotiation, and responsibility delegation. The more complex communication contents are the more time it will take for processing. Our investigations exposed that time critical communication between autonomous units or subsystems has to be coped directly between the real-time controllers. Even the third, physical, layer represents a communication channel, for example we could assume that certain tasks are triggered by the arrival of a physical part at a physical processing unit. Thus holons shall be able to communicate on all three layers.

3 Web Services

A Web Service represents a unit of business, application, or system functionality that can be accessed over the Web. Web Services are applicable to any type of Web environment, whether Internet, Intranet, or Extranet, whether with a focus on business-to-consumer, business-to-business, department-to-department, or peer-to-peer communication. A Web Service consumer could be a human user accessing the service through a desktop or wireless browser; it could also be an application program or another Web Service.

A Web Service exhibits the following basic characteristics:

- A Web Service is accessible over the Web.
- Web Services communicate using XML messages over standard Web protocols.
- A Web Service exposes an XML interface description.
- A Web Service is registered and can be located through a Web Service registry.

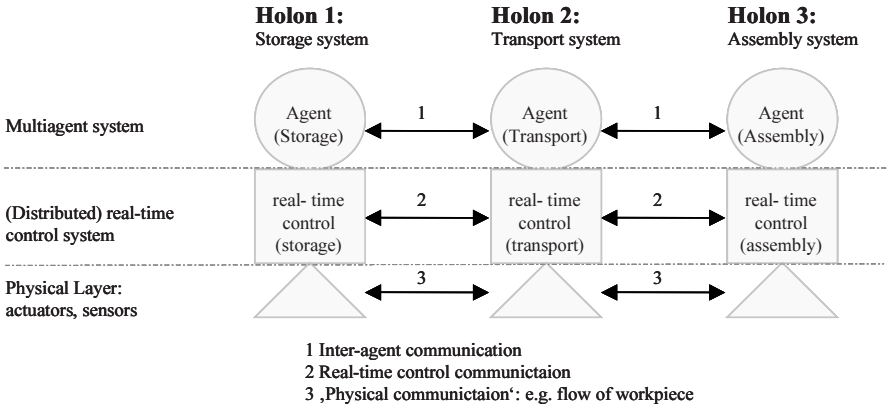


Fig. 1. Common architecture of shop floor holons

Figure 2 shows the Web Service Model where three kinds of roles and the performed operations can be identified. The roles are as follows [5]:

Service Provider: A service provider is the entity that creates the Web Service. Typically, the service provider exposes certain functionality in their organization as a Web Service for any organization to invoke. The service provider needs to do two things to reach the full potential of a Web Service. First, it needs to describe the Web Service in a standard format, which is understandable by all organizations that will be using that Web Service. Secondly, to reach a wider audience, the service provider needs to publish the details about its Web Service in a central registry that is publicly available to everyone.

Service Requestor: Any organization, any entity using the Web Service created by a service provider is called a service requestor. The service requestor can know the functionality of a Web Service from the description made available by the service provider. To retrieve these details, the service requestor does a search in the registry to which the service provider had published its Web Service description. More important, the service requestor is able to get the mechanism to bind to the service provider's Web Service from the service description and how to invoke that Web Service.

Service Registry: A service registry is a central location where the service provider can list its Web Services, and where a service requestor can search for Web Services. Service providers normally publish their Web Service capabilities in the service registry for service requestors to find a Web Service and then to bind it. Typically, information like company details, the Web Service that it provides, and the details about each Web Service including technical details is stored in the service registry.

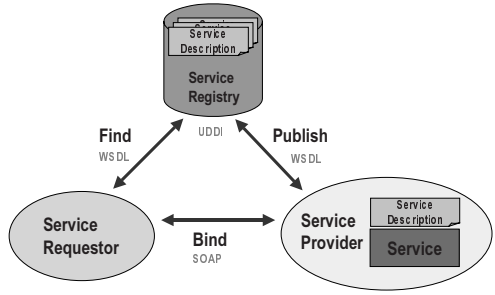


Fig. 2. Web service model

In the Web Service model there are three operations that are fundamental to make Web Services work – "publish", "find", and "bind". To achieve inter-application communication irrespective of the kind of language the application is written in or the platform the application is running on, the following standards for each of these three operations are defined:

- The WSDL (Web Service Description Language) is a standard that uses XML format to describe Web Services. Basically, the WSDL document defines the methods that are present in the Web Services, the input/output parameters for each of the methods, the data types, the transport protocol used, and the end point URL at which the Web Service will be hosted.
- UDDI (Universal Description, Discovery, and Integration) creates a global, platform-independent, open-framework to enable services to (1) discover each other, (2) define how they interact over the Internet, and (3) share information in a universal, Web-based business directory.
- A standard protocol for applications to bind to Web Services. The SOAP (Simple Object Access Protocol) is a lightweight XML mechanism used to exchange information between applications regardless of the operation system, programming language, or object model.

The application scenario is based on following assumptions:

- Production plans with sequential/parallel branches are given, consisting of linked production steps.
- The scheduling algorithm uses a short-term strategy, i.e. if a production step for an order is about to be finished, the next step in the production plan is scheduled. This minimizes the effort of re-scheduling complete schedules where large sets of production steps are allocated.
- Machines have input queues, where physical workpieces are stored. The queue somewhat decouples transport from machining, relieving the constraints of timely delivery of workpieces. As soon as a workpiece arrives in the input queue for a machine, no re-scheduling of the workpiece on this machine is possible.

The previously mentioned standards cover the first building block for the future service-based e-business automation called "Service description and transport binding". Other building blocks are [6]:

- Process description: The process describes the sequence or choreography of the operations the services supports. Additionally, this building block describes internal executable business processes that support the service's public collaborative process. Recently, there have arisen different XML-based standards like WSFL, XLANG, BPML, or XPDL. The first attempt to force only one international standard is the BPEL4WS (Business Process Execution Language for Web Services) released end of July 2002 – that remains to be seen if BPEL4WS will win out over the existing standards.
- Security: Security requirements include a combination of the following features: authorization, authentication, confidentiality, non-repudiation, and auditing. Any exchange of business information may require all, some, or no security features. To achieve the different security goals, emerging XML-based standards like XML Signature, XML Encryption, XKMS, SAML, or XACML were developed.

4 Shop Floor Holons Marrying Web Services Gives Birth to Shop Floor Services

Shop floor services marries both concepts that of holons and that of Web Services. It should keep the autonomy and smartness of holons and adds value by interfacing these with a new consolidating standard for distributed applications based on Internet technologies. This chapter will first present the shop floor service architecture, discuss then interconnecting shop floor services to control processes, and concludes with functional target areas and benefits.

4.1 Shop Floor Service Architecture

Figure 3 depicts the relationship between the building components. On the bottom layer there are sensors, actuators and according mechanics. These are controlled by the real-time control.

The real-time control layer encapsulates hardware and software for efficient, safe and reliable control of the machine or mechanical components beneath. That comprises a run-time environment for ready-made and customer programs and algorithms, which are offered to higher level components for configuration or parameterization. The HMS approach encourage the IEC 61499 [7] standard for distributed control but there are no industrial products available right now. Present systems rely on more or less IEC 61131-3 conform controllers. Both can be applied on this layer. It influences the way of communication with the top-level components. The IEC 61499 control concept bases on exchange of messages and events, which also means it is easy to interact with other information processing units. IEC 61131-3 offers messaging but in an inconvenient way. These capabilities were added quite late to the standard and were not a design principle from beginning. The established standard for accessing data in currently used controllers is OPC, OLE for Process Control [8].

The top layer offers the basic mechanical functionalities with their according real-time control extended by information processing capabilities for smart behavior and

integration on system level. These components could be derived from the agent's paradigm for example knowledge base, rule engine, scheduler, execution monitor, and communication patterns for different negotiation or auction types. The high level components could be scaled from simple production data access to full intelligent software agent capabilities [9].

The big deal of the shop floor service approach is to offer all the above listed features in a standardized way as Web Services to enterprise information systems and manufacturing automation units for automation coordination and vertical integration. It allows the homogenous integration of even single sensor information into holistic process automation. Holistic process automation is here meant as integration of typical business processes like order acceptance, supply chain coordination, payment handling with shop floor automation components.

For the application of Web Services we need three additional software components, a Web Service description, a web server and a Web Service run-time environment.

- The Web Service run-time unit provides an environment for execution of Web Services. In particular, it maps incoming SOAP requests to the appropriate component or method, so it is responsible for launching the requested functionality/-ies. At the same time it collects results and wraps them into SOAP responses. The run-time environment also formulates SOAP requests if other, non-local Web Services are involved on demand of underlying functions.
- The web server component is the main gateway for SOAP requests to be received by the service provider. SOAP messages are represented in XML and are transported via HTTP. A common web server communicates via HTTP protocol and normally listens to port 80. That implies that SOAP requests can pass through nearly all firewalls – accessibility from everywhere. The web server receives SOAP requests and forwards them to the Web Service run-time component.

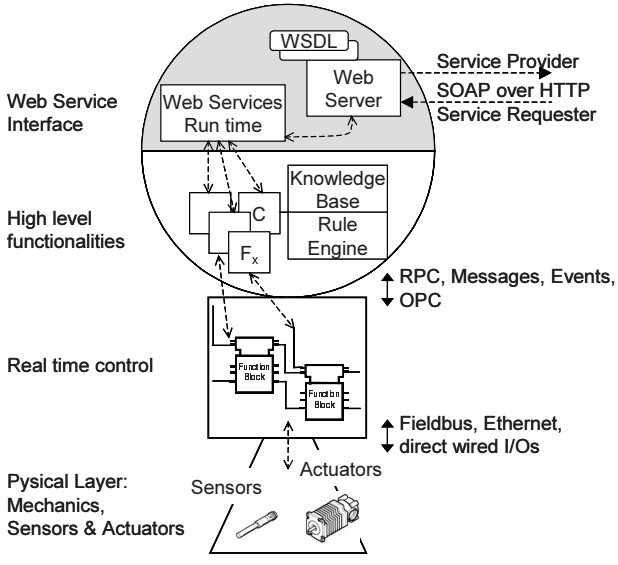


Fig. 3. Shop floor service architecture

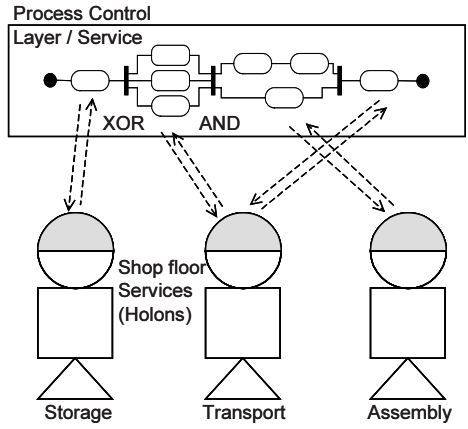


Fig. 4. Vertical shop floor holon integration

- A key thing, which has to be noticed, is that the WSDL file, the Web Service specification file, resides on the web server at the device and could be maintained according to the availability of particular functions. The UDDI server maintains a valid copy of the WSDL file. Service requestors discover Web Services through the UDDI registry.

A shop floor service can act either as service provider or service consumer.

4.2 Shop Floor Services Orchestration for Process Control

The preceding subchapter introduced a single service on the shop floor. To produce a product or part proper-sequenced services are needed. Figure 4 shows an abstract layer containing all information of the particular manufacturing process, the process control layer. The process control specification can itself be a kind of high-level shop floor service that constitutes through the orchestration of several basic services. A process control service can be requested to initiate the accomplishment of a particular manufacturing process. In the process control layer the sequence of all required shop floor services are specified. For process definition there are several endeavors of standardization running, to name the most important organizations: Business Process Management Initiative (BPMI), Workflow Management Coalition (WfMC) and the Object Management Group (OMG) who are converging in defining process definition languages.

The separation of process sequence information and process execution entities emits two supporters of flexibility. If a new product or a product change is required a new process control service has to be designed. That does not interfere the shop floor services itself because they only offer their functionalities. On the other hand process definitions allow deciding in process which shop floor service to request next dependent on global process information.

Figure 4 sketches an example process where parts are requested from storage, then transported to an assembly station and afterwards brought to an e.g. packaging station. For transportation three automated guided vehicles are available. Which one will be used is decided during the processes run-time depending on the vehicles current workload and availability.

4.3 Building Holarchies

Design principles for software and complex systems require transparent structuring of comprehensive functionalities. Holonic induct open hierarchies, holarchies, where entities of arbitrary complexity integrate with others to higher level of complexity.

Figure 5 shall give an idea of clustering shop floor services in a compound to fulfill a sub process. It is important to note that such a holarchy is dedicated to a certain task or process. When same resources are involved in different processes they will belong to several holarchies or compound shop floor services. It is easy to change a holarchy in changing the process definition that includes the different types of involved services as well as their sequences and triggers.

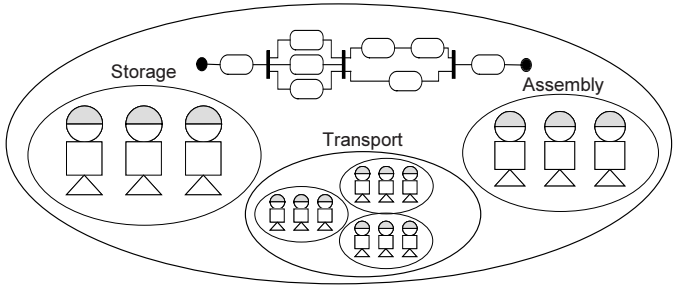


Fig. 5. Shop floor service orchestration

4.4 Functional Target Areas

The above mentioned features mostly address the problem of high flexibility, high volume high variety production, facilities for automated reconfiguration of processes, etc. There are a couple of additional outstanding benefits:

- Low cost access of automation devices from everywhere. Nearly no restrictions caused by firewalls (SOAP over HTTP).
- Instant, remote and homogenous access of production and machine data for monitoring, logging, analysis, visualization and diagnosis.
- Remote services maintenance; e.g. up- and download of control programs and algorithms, parameterization, service configuration, etc.
- Scaleable local intelligence
- Seamless integration in business processes, in particular with business software. In other words emancipation of business services with intelligent mechatronic automation units.
- Forces stringent component oriented design and engineering of manufacturing systems.
- Universal interfaces and service enactment
- Robustness on operative level through autonomous, intelligent units

5 Reference Implementation at the Odo Struger Lab Testbed

To convince industry it is not always the most important thing to come up with closed theoretical elaborations. To gain attention and acceptance it is fundamental to present running prototypes. At the University of Vienna a testbed for holonic and distributed control was built up the last 3 years, which is a great chance to proof the proposed concept.

Figure 6 illustrates the testbed and the system composition. It consists out of three subsystems a storage system, a transport system, and an assembly system. Each is composed out of a set of sub-systems or entities modeled as sub-holons. The service access points of each system are the proxy holons, e.g. the transport manager holon, that can receive job assignments. All holons can be accessed through their Web service to e.g. collect data, query states, inquiry work load, etc. Inside a system holons coordinate their behavior through negotiations. A superposed business process requests the required tasks from the system.

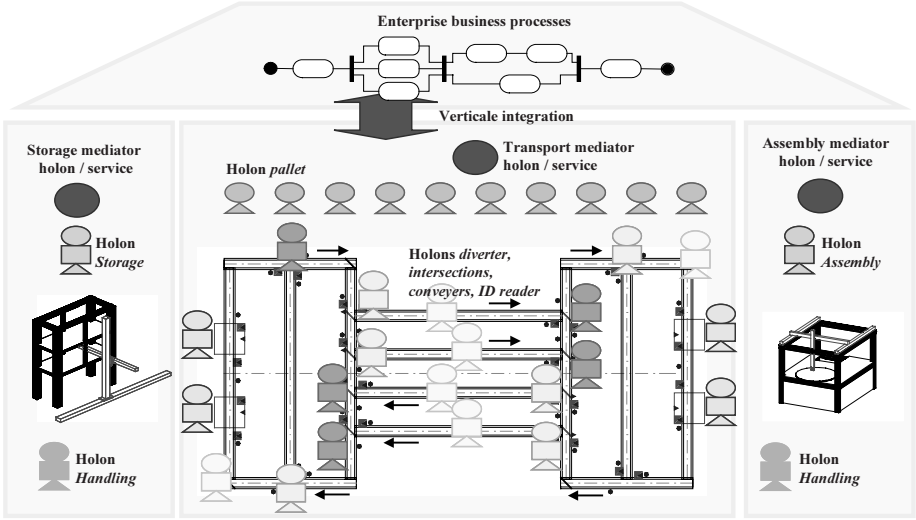


Fig. 6. Reference implementation

Finally considering the applied technologies. Holons' real-time controllers are IEC 1131 PLCs or self-developed embedded IEC 61499 controllers. For the agent realization the ZEUS framework from British Telecom allowed rapid agent implementation. To run the business process a workflow engine was extracted from the open-source project OFBiz – www.ofbiz.org. Processes are defined in the XPDL XML Process Definition Language from the WfMC Workflow Management Coalition – www.wfmc.org.

6 Conclusion

The prototype reveals the expected benefits of easy process modification respective maintenance and robust task execution allowed through the deployed holonic paradigm on the shop floor. It proofs the concept of integrating holonic shop floor control with business processes via Web Services. It could be seen as a migration step towards full the holonified enterprise.

References

1. A. Koestler, The ghost in the machine, Fritz Molden Verlag, Bern, 1968.
2. M. Fletcher, E. Garcia-Herreros, J. H. Christensen, S. M. Deen, and R. Mittmann, "An open architecture for holonic cooperation and autonomy," proceedings of Database and Expert Systems Applications, 2000.
3. J. H. Christensen, "HMS: Initial Architecture and Standards Directions," proceedings of 1st European conference on Holonic Manufacturing Systems, University of Germany, Berlin, 1994.
4. C. Dutzler, K. Glanzer, and G. Zeichen, "Hochflexible Steuerungsarchitektur für Montageprozesse auf Basis verteilter holonischer Systeme," proceedings of SPS/IPC/DRIVES 2002, Nürnberg, Germany, 2002.
5. M. Hendricks, B. Galbraith, R. Irani, J. Milbery, T. Modi, A. Tost, A. Tousaint, J. Basha, and S. Cable, professional Java Web Services. Birmingham, UK: Wrox Press Ltd, 2002.
6. K. Fürst, T. Schmidt, and G. Wippel, "Enabling Collaborative Engineering with Web Services," presented at Conference on Inteligent Engineering Systems, Opatija, Croatia, 2002.
7. R. W. Lewis, Modelling control systems using IEC 61499. London: The Institution of Electrical Engineers, 2001.
8. F. Iwanitz, J. Lange, OLE for process control, Hüthig, Heidelberg, 2001.
9. G. Weiss, Multiagent Systems - a modern apporach to Distributed Artificial Intelligence. Cambridge, Mass.: MIT Press, 1999.

Proposal of Holonic Manufacturing Execution Systems Based on Web Service Technologies for Mexican SMEs

Luis Gaxiola[1], Miguel de J. Ramírez[2], Guillermo Jimenez[3], and Arturo Molina[1]

[1]ITESM Campus Monterrey, Integrated Manufacturing Systems Center, Av. E. Garza Sada 2501, 64849, Monterrey, Nuevo León, Mexico
{luis.gaxiola, armolina}@itesm.mx
http://csim.mty.itesm.mx
[2]ITESM Campus Monterrey, Mechatronics and Automation Department, Av. E. Garza Sada 2501, 64849, Monterrey, Nuevo León, Mexico
miguel.ramirez@itesm.mx
[3]ITESM Campus Monterrey, Informatics Research Center, Av. E. Garza Sada 2501, 64849, Monterrey, Nuevo León, Mexico.
guillermo.jimenez@itesm.mx

Abstract. This paper presents a novel approach to Manufacturing Execution Systems (MES) using Web Services technologies and protocols to integrate the information from different sources to different destinations inside a holonic environment. The approach is addressed to integrate systems in small and medium manufacturing enterprises of Mexico when the use of commercial management software is reduced due to high cost and poor adaptability to SME from developing countries. A design of a MES Holon is presented in order to be flexible to adapt on this kinds of SMEs. Based on the results derived from this validation process of the model, whose first stage is expected to be finished in September, 2003, the goal is to extend and transfer this model to three Mexican SMEs to support its technological modernization.

1 Introduction

1.1 The Role of Manufacturing Execution Systems in Production Environments

The concept of MES (Manufacturing Execution System) has acquired a huge importance in today's modern manufacturing industries. As a matter of fact, MES has a direct relationship with MTO (Make to Order) industrial environment (nevertheless, this fact doesn't discard MES usage in any other industrial environment). Some characteristics of MTO environments, such as the existence of an initial product specification basis for product's customization, and the need of acquire the ability to apply fast changes to a customizable basis in order to conform last-minute customer's order changes [1] make this systems specially suitable and profitable for MTO. MES is a relatively new concept, introduced in the 1990's, and different authors have given numerous definitions about it. Some of the most complete of them, have been developed by MESA (acronym for Manufacturing Execution Systems Association; 1997) and Mintchell (2001). According to MESA, Manufacturing Execution Systems deliver information that enables the optimization of production activities from order launch to finished goods. Using current and accurate data, MES guides, initiates,

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 156–166, 2003.

responds to, and reports on plant activities as they occur [2]. Under Mintchell's point of view, MES is a software-based application that grew up in the realm of planners to help schedule optimum production runs, manage inventory, handle regulatory databases and information, and other manufacturing planning chores [1].

Considering all reviewed definitions, MES has the following two characteristics:

- It provides real time information about what is happening in the shop floor, for managers (under a strategic approach), and for direct operation workers (under a purely operative approach).
- It is an information bridge between Planning Systems used in Strategic Production Management (such as ERP) and Manufacturing Floor Control SCADA (Supervisory Control and Data Acquisition). It links the Manufacturing Information System's layers (Strategic Planning and Direct Execution) through the adequate on – line managing and control of updated information related with the basic enterprise resources: people, inventory and equipment [3].

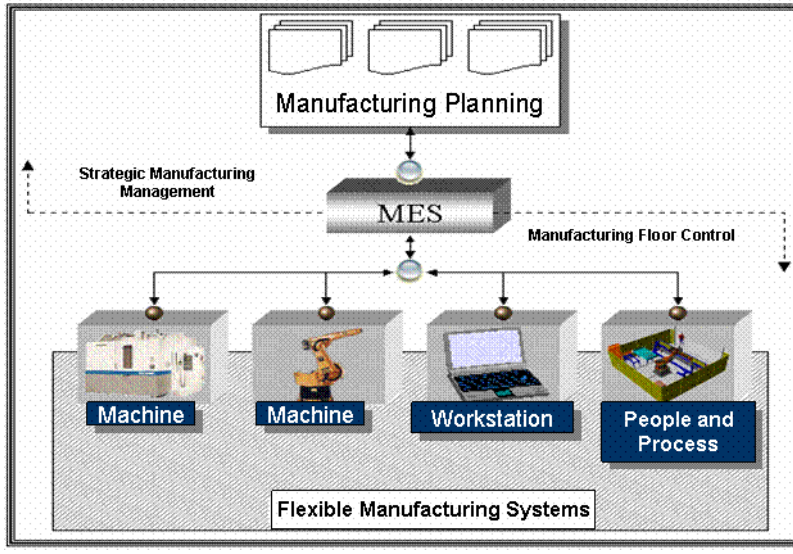


Fig. 1. MES role as an information bridge between *Strategic Manufacturing Planning* and *Manufacturing Floor Control*

The enormous importance acquired by MES resides, in a significant percentage, on its functionalities (implying people, inventory and equipment) and their interaction with the compounding elements of the industrial plant environment. Authors and organizations, such as McClellan and MESA have given full and detailed descriptions of these functionalities, independently of the industrial application branch of MES. Some other authors, like Kaufmann [4] and Schenker [5] have emphasized some special functionalities and technologies that suits specifically to Semiconductor Industry and general SMEs (Small and Medium Enterprises). Some other authors, like Fukuda [6] and Cheng et al [7] have developed important MES-next-generation architecture models. Nevertheless, none of them have proposed a radically different

or non related functionality beyond MESA's basic definition. MESA has proposed a hierarchies distinction for MES functionalities. This hierarchy is divided in two stacks: *Core Functions* and *Support Functions*. Core Functions include Planning System Interface, Data Collection, Exception Management, Work Orders, Work Stations, Inventory / Materials and Material Movement. On the other hand, MES Support Functions are listed as Genealogy, Maintenance, Time and Attendance, Statistical Process Control, Quality Assurance, Process Data and Documentation Management. Interwave Technology forecasted some important trends related to MESs immediate, mediate and long term future, concerning to manufacturing enterprise collaboration and supply chain performance [8].

2 Holonic Theory Applied to Manufacturing Systems

A powerful reason motivating this research is the fact that integrated MES's (integrated suite of application systems) are sometimes regarded as monolithic, insufficiently configurable, and difficult to modify [7]. These MES's are designed inside the frame of hierarchical computer control, which wouldn't be suited in SMEs environment. Related with this, Duffie *et al.* (1986) propose a non-hierarchical (heterarchical) system. The core of this research centers on a series of design principles that produce a system of cooperating, autonomous entities. These design principles have been applied for the heterarchical control of a manufacturing cell [9].

It is extremely important to remember that, besides the external events, each SME is constantly immersed in an instable production environment; due to their small size, each SME possess distinctive characteristics such as a low complexity management, low costs for manufacturing activities and a high dynamism. Aiming especially at this last characteristic and to the mentioned non-stable production environment, the necessity of quick and full changes to overcome it successfully arises. This necessity can be satisfied through the application of modern manufacturing paradigms, such as *Holonic Theory* [9]. Basic concepts related with the term *holon* were used initially by Koestler (1967) as a way to describe an identifiable and indivisible part of a whole system that possesses individual identity and is composed of sub-ordinate parts. One of these concepts, with a great relevance, is *Holarchy*, described as a stable, self-contained system consisting of holons as building blocks. Many applications and interpretations of holonic concepts can be found in literature, comprising numerous human disciplines. It is important to remember too, that the strength of holonic organizations (holarchy), is that it enables the construction of extremely complex systems that are nonetheless efficient in the use of resources, highly resilient to disturbances (both internal and external), and adaptable to changes in the environment in which they exist. All these characteristics can be observed in biological and social systems [7]. This research pretends to integrate to Manufacturing Execution Systems all the mentioned benefits that holonic structures grants. Mentioned characteristics, in the fields of organizational, technological, productive and information systems- are not desirable but essential for all organizations survivorship, especially for those facing a day-to-day highly changing environment, as is the SMEs case.

Important developments and theoretical models have been developed to achieve an Agile Manufacturing Environment. One of these models has been developed by

Rahimifard [10], which consists, mainly in a practical representation of holonic manufacturing systems, focused to SMEs and based on three research areas, namely a holonic information network, a holonic production planning and control structure, and holonic manufacturing workstations. A significant contribution of this work is a conceptual holonic representation of an IT supported SME, with three main holons: an executive holon, in charge of the ultimate decision-making process within the company; the business holon that covers administration activities such as order processing, finance, costing, process planning and scheduling etc.; and the manufacturing holon involving the implementation and monitoring of the production plans produced by the business holon.

3 MES Model in SMEs Using Holonic Approach

In order to achieve a good comprehension of the holonic nature of the presented MES model, it is important to mention some significant facts. A typical Mexican SME (especially in the metalworking sector), from a physical and communicative approach is mainly visualized as an operation floor with automated but not integrated machinery, such as lathe CNCs, mill CNCs, CMMs, etc. In addition, this typical Mexican SME has specialized software for the core operative activities, such as CAD, CAD/CAM, etc., and for administrative and support activities, like accounting, personnel managing, maintenance schedules, etc.

One of the pretended flexibility characteristics of this development aims toward the adaptability among the MES Holon and the integrated or isolated elements described previously. This adaptability leads to the generation of the following two mutually excluding configurations in the operation floor:

a) An operation floor comprising partially isolated elements. It is characterized by a "Stand-Alone" MES (See Fig. 2 a), which is used under an informative approach, and acts just as a data collector from the equipment's information interfaces and operative activities, in order to process, update, record and deploy strategic information to the involved users. In this case, not all its designed functionalities work, just those informative such as Scheduling Simulation, Time and Attendance, Quality Management, Performance Analysis and as a work order's pricing tool. This functionalities and the information generated by them don't have any supervisory control character over machines. This is a low financial and technical investment option. The machines and the computers containing specialized software are communicated with the MES through a holonic communication interface (HCI) for each one.
b) On the other hand, our designed MES model can be seen, based on Rahimifard's concept (2002), as a holonic element inside a holonic information system (HIS), as depicted in Fig. 2 b [10]. That is the reason why this Figure shows the integration among three main holons: Executive Holon, MES Holon (the proposed Model) and Operation Floor Holon. When it is integrated in a HIS, all it's designed functionalities are set aside, due to the fact that some of them have regulatory character over the Operation Floor Holon and others have just informative character concerning to the entire holons coupled to the HIS. This described condition can be considered as a MES full integration to HIS. The

holon in which this research is based, as it was said, is the MES Holon. This holon is in charge of all manufacturing administration activities (but not of ultimate decision-making process, because this function is done by the Executive Holon), being some of them ERP's info receiving (or Info Receiving directly supplied by the Production Planner), Scheduling Simulation, Time and Attendance, Control of Materials, Process Routing Management, Performance Analysis, Quality Management and, as a link to the Operation Floor Holon through the HIS, a Data Collect and Supply Interface. The main holons and the machines included inside Operation Floor Holon are integrated to the HIS through a holonic communication interface (HCI) for each one.

In both configurations, the attention and efforts of this research are focused to the MES Holon. It is important to remark the fact that the adopted configuration depends on SME's decision concerning to financial investments and technical issues on communication networks, HIS interface software required for each integrated machine, desired integration level, etc. The Holonic character of the proposed MES model is bestowed and explained by the following characteristics of the MES and the system in which is immersed:

- The first one is the *heterarchical relationship* among Executive Holon, MES Holon and Operation Floor Holon in the case they are coupled to HIS (Fig. 2 b).
- As depicted in Fig. 2 a, *MES is a self contained entity*, able to exist coupled or not to a HIS. When MES is not coupled to a HIS, it has just informative character (historic and real time) and just its designed informative functionalities work. On the other hand, when the three main considered holons are coupled through a HIS, they act as *cooperative* and *integrated* entities. Under this scheme, all designed MES functionalities work.

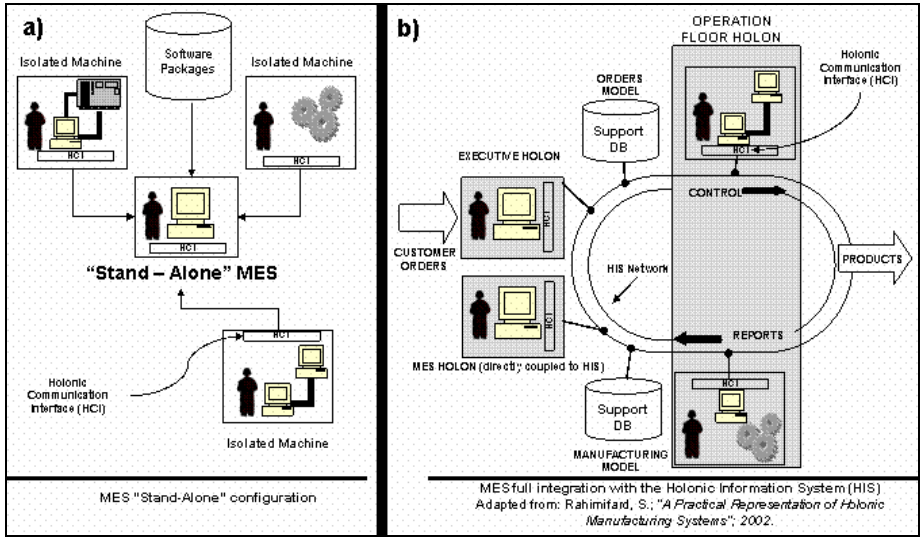


Fig. 2. a) *MES "Stand-Alone"* configuration: *MES* acts just under an informative approach. b) *MES* full integration to the *HIS*

3.1 Designed Functionalities for MES Holon

As can be seen in Figure 3, the Model for MES Holon Functionalities and their relationship through information flows is shown in a conceptual form. These functionalities are:

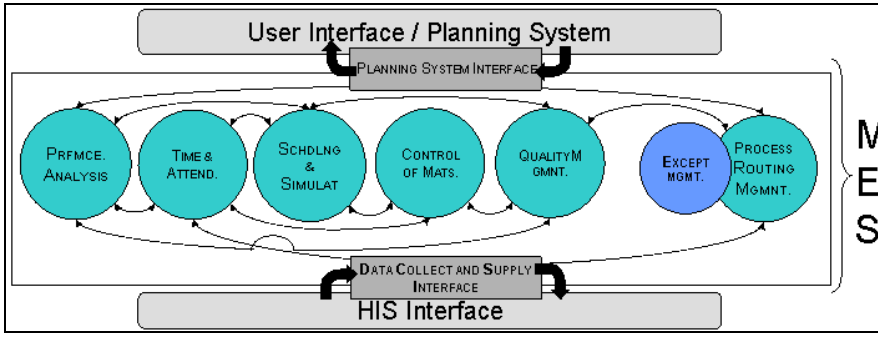


Fig. 3. *General MES Holon* Model Proposed for *SMEs*

- **Process Routing Management.** As inputs, it receives a State of Resources report (this report, when displays critical numbers, acts as a trigger for production stop). Besides that, it receives the basic information related with Work Orders, including Order's Priority Hierarchy, and some other features like *Size*, *Product ID. Etc.* Some other information is required here, like Previous Order Progress, in order to plan the Manufacturing Floor production sequence. It's outputs are directed to the FMS – Operation Interface. This outputs are Material Handling and Material Management Orders (for general Material Control) and the required list of CNC, Robot and General Machine Programs, in order to be retrieved by the correspondent controllers.
- **ERP's Info Receiving or Planning System Interface**. This functionality acts as an information gate, without significant info processing operations. Strategic information received includes Tentative Due Dates, BOP's (Bills of Processes, according to the received Work Order), BOM (attached Bill of Materials for the Work Order) and, besides that, it receives the approvals for the MES Scheduling Simulations (see "Scheduling Simulation" Functionality) from the Production Planner or Manager (moreover, from the Planning System) as a trigger for the required operations in order to finish a Work Order.
- **Scheduling Simulation.** It acquires all the information related with the updated state of resources (such resources are labor force, materials and equipment availability) and, through a simulation procedure, it delivers forecasted state of resources after work order completion and finishing time. This information is sent to the Planning System Interface, with a "for approval" status.
- **Exception Management.** This functionality is considered as a subsection of Process Routing Management. It manages all pre-programmed routines and algorithms required when there is a non-normal condition in the system, such as material scarcity, a down machine, status or date change in a work order, among others.

- **Performance Analysis.** It receives information from the Time and Attendance during working sessions, from Material Control as a report of finished goods and reworks, from Quality Management as dimensional or featuring non conformities and, from Process Routing Management as the forecasted Work Order time and the real processing time. All this data is processed to compute a Performance Analysis. This Performance Analysis is computed order by order and globally, after certain periods of time.
- **Quality Management.** It receives information from Quality Control Variables from CTQ (Critical to Quality Variables) -in-Product and in-process Measurement Devices. Its outputs are sent, mainly, to the Performance Analysis Functionality and to the user's interface through a Global Quality Report.
- **Time and Attendance.** Acquires data related with the Human Resources Availability, in order to generate the adequate updated information related with Operative Labor Force and it's performance, necessary to worker's *dossier* and, mainly, to the Scheduling Simulation.
- **Control of Materials.** It plays a vital role. This Functionality doesn't only counts and tracks inventory, but is in charge of all equipment related with Material Handling and Movement. Its inputs are Order's Process Routing Order Features (like *Work Order size*, *Product ID, etc.)*, and Direct and Non Direct Materials state of resources. Its outputs are, mainly, Material Handling Equipment (AGV's, conveyors, robots, pallets) execution orders, Warehouses Material releasing permissions and Reports of State of Resources (Direct and Non Direct Materials).
- **Operation Interface or Data Collect and Supply Interface.** As the ERP's Info Receiving or Planning System Interface, this functionality acts as an information gate, without significant info processing operations. It's inputs are Digital signals and variables, coming from different sensing and status devices, switches, etc. All of this signals are just derived in a standard format (as output) to the functionality that needs them, like Quality Management, Control of Materials, Time and Attendance and Process Routing Management.

3.2 Implementation Tools

Manufacturing enterprises are complex environments that require managing many interrelated resources (e.g., human, supplies, machines, etc.). The proposed system will be a building block in whole the spectrum of manufacturing management systems, particularly in its integration to MRP II/ERP and quality control systems, which includes its link with the supply chain. The technologies used in the implementation will easy its deployment in different manufacturing environments with different existent software and hardware platforms.

System integration has been approached by several proprietary and open technologies [11]. Remote procedure call, message based and peer-to-peer technologies have been successfully used to integrate common platform systems [12]. However, multiplatform systems have been difficult. A recent proposal to multiplatform integration is to use Web Services technologies and protocols. Web services use standard text based protocols for message description. In a Web service, message description specifies the operations that can be requested, the necessary parameters and the type of results produced. Message parsers are implemented in

different programming environments and the corresponding operations executed. The result is translated to a textual representation that can be parsed using any other platform. Such platform independent specification provides a mechanism that can be implemented in different programming languages. The particular protocols used by manufacturing units are translated to standard protocols that can be handled in a uniform way by different implementation environments.

In order to reach the implementation of the MES Holon inside of a holonic environment, it is necessary to have an architecture with the characteristics mentioned above. Figure 4 show a proposal frame using Web services mediators as HCIs to implement the MES Holon. To simplify the integration of systems from different vendors, it will be required to develop interoperability protocols; these protocols will be designed using XML and SOAP; WSDL will be used to simplify integration with specific APIs that offer the different systems being interconnected.

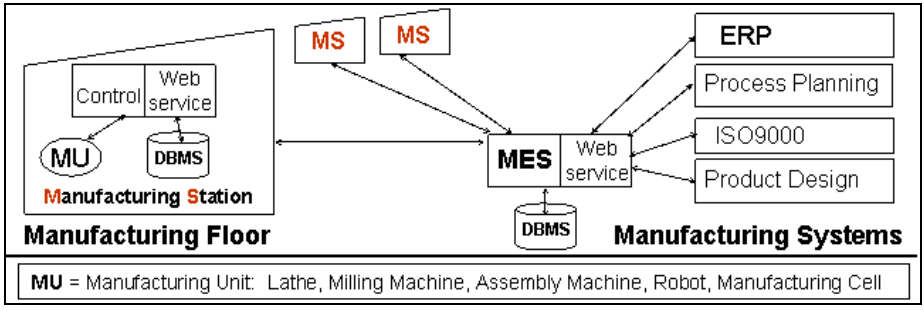


Fig. 4. *MES* Architecture using *Web Services mediators*

4 Application of Proposed MES Model

Micro, small, middle and big industry are or great importance in the economy and employment both in industrialized countries and those with lower development level, which is the case of Mexico. According to Economical Census, carried out by Mexican government at the end of the 1990s decade, micro, small and middle companies constituted the 99.6 % of Mexico's industry Mexican companies are immersed in macro-economic surroundings, national surroundings that are not stable, therefore, the investment patterns cannot be stable. The priorities of these companies are not to have the latest technology, but to maintain a healthy economy.

4.1 Use of Information Technology (IT) in Mexican SMEs

Micro, small, middle and big industries are of great importance in the economy and employment both in industrialized countries and those with lower development level, which is the case of Mexico. According to Economical Census, carried out by Mexican government at the end of the 1990s decade, micro, small and middle companies accrue to 99.6 % of Mexico's industry. Mexican companies are immersed in macro-economic surroundings and national surroundings that are not stable; therefore, the investment patterns cannot be stable. The priorities of these companies are not to have the latest technology, but to maintain a healthy economy.

The core software application for the Mexican companies is the management packages, which amounts 64% of national market. In this classification they are from simple packages like Windows, to vertical applications like NOI, COI, HAMMOCK, West Wind, Adam, among others. Customized Software Development (CSD) is the second software application more important for Mexican companies, due to its adaptability and low cost in comparison with robust solutions like ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) or SCM (Supply Chain Management) and because sometimes is the company's IT department which itself carries out the required software developments, in order to downsize costs and due to their knowledge on the own specific company's processes. Nowadays, there is a trend for Manufacturing Management software suppliers to cover and satisfy demands on this technology for big industry in Mexico, which technological investment in this area is less than 2% [1]. Therefore, the current strategies are looking downstream and trying to fit the needs of medium, small and micro companies. From this we can derive that there is a need for specialized but configurable low cost software. Configurable and low cost MES software could be a helpful information tool for SMEs, to improve their productivity and competitiveness. This implies that no modules need to be sub profited (as it could happen with the use by middle to down companies, of software designed for large companies).

4.2 The Application of MES in Mexican SMEs

As a research initiative to validate this Model, it is planned to apply it to three representative SMEs located in the metropolitan area of Monterrey, Mexico, at the end of 2003. These three enterprises belong to the metal-mechanic and plastics industrial branches. General information about the enterprises is described in Table 1. It is planned to customize this Model during it's implementation, according to each SME situation or condition. The main conditions to be considered into this customization are: *human resources*, *economical resources* and *industrial operation scheme.* The strategy defined in this technology transfer project for the implementation is listed below, based in proposed MES implementation strategies from MESA, Carnegie Mellon and Cheng et al:

1. Perform Manufacturing assessments for key operations
2. Apply diagnostics and tools during assessment phase.
3. Identify the Manufacturing pain and the potential improvement areas
4. Develop a Thorough Understanding of the Opportunity Areas
5. Link the areas of opportunity to corporate benefits.
6. Re-Design individual updated model according to each enterprise's situation or condition (human resources, economical resources and industrial operation scheme).
7. Begin Pilot MES Project and Assess Results Against a Baseline (this implies an iterative assessment made by end users and designers).
8. Make the Final Validation or Releasing.
9. Close Project's Documentation (this process is going to be iterative and permanent in each stage of the project) [5] [7] [13].

5 Conclusions

A MES Model for Mexican SME Industry with its functionalities, is presented. This proposal will allow to establish two integration configurations for the equipments and main manufacturing resources, according to the financial and technological investment of each SME. The use of Web service technologies is proposed in order to interconnect different Manufacturing and Production Planning Units getting flexibility in order to customize the MES to SME´s necessities. The low cost is reached using standard PC platform. The project will have a great impact on the technology modernization of SME´s, because actually this kind of enterprises has not access to this technology due the high cost and the complexity of the commercial software.

Table 1. General information about SMEs in which the proposed MES Model is planned to be applied

FEATURE	SME 1	SME 2	SME 3
Industrial branch and Market	Metal – Mechanical industry, dies fabrication.	Metal – Mechanical industry, Mining, Iron and Steel.	Rubber and Plastics, Toys and Domestic Electrical Appliances.
Human Resources Specialization Level and /or Quantity	Operation Workers: 30 Designers: 4 Programmers: 2	Engineers: 2 Technicians: 22 Buyers: 1 Sales: 2	Specialists in Plastic Mold Design.
Core IT and/or Informatics Technologies	Designing Software (AutoCAD, Unigraphics, Striker Die Maker System).	Designing Software (AutoCAD).	Designing Software (AutoCAD)
Operation Scheme	MTO (Make to Order)	MTO (Make to Order)	MTO (Make to Order)
Main Customers	• GE Aircraft Engines, Industrial Systems and Power Systems • Lithonia Lighting Mannesmann Sachs Metalsa • Hamilton Beach	• Mexican Federal Electricity Comission. • Imsa Signode • Rassini • Stabilit	• Mattel • Mabe • Thomas & Betts • RTC Industries • Sinclair & Rush

Acknowledgements. The research reported in this paper is part of a Research Chair in Manufacturing of ITESM titled "Design, Manufacturing and Integration of Reconfigurable and Intelligent Machines". The authors wish to acknowledge the support of this grant in the preparation of the manuscript.

References

1. R. Meneses: Diseño e Implementación de un Sistema MES flexible y de bajo costo para la mediana empresa (Design and Implementation of a Flexible and low cost MES System for medium enterprises). MSc. Thesis, in Spanish, ITESM, Monterrey, Mexico (2002) 23-32
2. MESA (Manufacturing Execution Systems Association): The Benefits of MES: A Report from the Field. USA (1997) 1–5
3. M. McClellan: Introduction to Manufacturing Execution Systems. MES Conference & Exposition, June 12-14; Phoenix, AZ, USA (2000) 3–11
4. T. Kauffmann: The Paradigm Shift for Manufacturing Execution Systems in European Projects and SEMI Activities. Semiconductor FABTECH, 8th Edition. Fraunhofer Institut Manufacturing Engineering and Automation, Stuttgart, Germany (1997) 17–25
5. F. Schenker, D. Cilia: Dynamic Scheduling and Simulation in a Job Shop Environment. Project Notes from Carnegie Mellon Software Engineering Institute and U.S. Department of Defense. Pittsburgh, PA. USA (2001)
6. Y. Fukuda: Production System Modeling and Manufacturing Execution System. Project Notes from JOP / Production System Modeling Technical Committee, Department of Industrial and System Engineering, Hosei University, Japan (1999)
7. F. Cheng, S. Wu, C. Chang: Systematic Approach for Developing Holonic Manufacturing Execution Systems. IECON'01: The 27th Annual Conference of the IEEE Industrial Electronics Society. Institute of Manufacturing Engineering, National Cheng Kung University. Taiwan R.O.C. (2001) 261–266
8. Interwave Technology, Inc. Next – Generation e – Manufacturing Solutions: Myths, Morphs and Trends. "The White Paper Series". USA (2001) 8–9
9. K. T. K. Toh, S. T. Newman: The future role of DNC in metalworking SMEs. International Journal of Prod. Res.,. 1996. Department of manufacturing Engineering, Loughborough University of Technology, Loughborough, Leics., LE11 3TU, UK, Vol. 34, No. 3, 863–877
10. S. Rahimifard: A Practical Representation of Holonic Manufacturing Systems. Fifth IFIP/IEEE (BASYS'02), September 25-27, 2002. Cancun, Mexico (2002) 323–330
11. D. S. Linthicum: Enterprise Application Integration, Addison-Wesley, (1999).
12. W. L. Oellermann: Architecting Web Services, Apress, (2001).
13. MESA International: White Paper Number 7. Justifying MES: A Business Case Methodology. USA (2000) 1–4

Secure FIPA Compliant Agent Architecture Draft

Tomáš Vlček[1] and Jan Zach[2]

[1]Czech Technical University in Prague, Czech Republic
vlcek@labe.felk.cvut.cz
[2]CertiCon a.s., CAK Prague, Czech Republic
zach@certicon.cz

Abstract. This paper deals with introducing general and open security mechanisms into the FIPA (http://www.fipa.org) compliant multi-agent architectures. A general overview of possible security threats and security services (countermeasures) is given first; then additional requirements, our design decisions and points of view concerning fundamental services (confidentiality, authentication, authorization, and integrity) are outlined. Further mentioned services, based on the fundamental ones, can be without concern added to the system.

1 Introduction

This paper deals with analysis of security threats and general requirements for security infrastructure intended for FIPA compliant multi-agent communities and to outline a potential solution for fundamental services. As fundamental services we consider confidentiality, authentication, authorization, and message integrity service.
One of the most principal requirement to the proposed solution is that the solution must remain FIPA compliant which means that every FIPA compliant agent should be able to interact with agents using the proposed security architecture even without any knowledge of that architecture and mechanisms; agents demanding for security can, of course, refuse to communicate with non-authenticated agents, reject non-encrypted messages, etc.

The proposed security mechanisms are intended to utilize existing FIPA agent platform mediators as Agent Management System (AMS), Directory Facilitator (DF) [4] to make these mechanisms transparent as much as possible and to avoid increasing number of facilitators necessary to run agent platforms; moreover the new facilitators implementing security mechanisms would in a great extent double that already existing functionality.[1]

[1] One of the consequence of introducing security mechanisms to agent platforms is that mediators will be able to provide more reliable services unlike current: “directory facilitator cannot guarantee the validity or accuracy of the information that has been registered with it” [4], etc., which flows from lack of such services.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 167–178, 2003.

This proposal is aimed mainly to "static," i.e. non-mobile, agents but it should be possible to extend the architecture about (rather non-trivial)[2] support of security services for mobile agents easily.

Frankly, the intention of this proposal is a consideration of design of generic, scalable, reliable, and widely-applicable security architecture, approached from the holistic point of view, for distributed and multi-agent systems. However, the goal of this draft is not to reinvent a wheel but rather to employ well-known and proved mechanisms.

Selection of used crypto-algorithms and protocols is based on the underlying survey and overview of issues of network security stated in [14].

2 Concise Security Threats Overview

Attacks on communication among agents can be roughly divided to passive and active attacks,[3] a brief description follows:

2.1 Passive Attacks

Passive attacks do not modify the data stream being attacked; they are based on eavesdropping and data stream monitoring with consecutive traffic analysis.[4] Attacks of this kind are aimed to revealing the plain content of messages, getting additional information as secret keys, etc. Detection of such kinds of attacks is almost impossible so that their countermeasures address rather their prevention as data encryption is, etc.

2.2 Active Attacks

Active attacks, in addition to the passive attacks, involve modification of data stream, creation of fake streams, and/or disabling availability of an agent or service.

---

[2] The problem of security of mobile agents is that a movement of an agent means a movement of its source/executive code and/or all agent's data to the selected target agent platform which is then responsible for execution of the agent's code. This mechanism brings myriads of potential threats – the agent can attack the platform, the platform can attack the agent, the agent another agent, etc. Authors of this paper are not aware of a general and good enough satisfactory solution.

[3] Except attacks to communication channels there are attacks to the inherent logic of security systems including software bugs and various vulnerabilities which may not directly concern attacks to the communication channels and we do not deal with them in this paper.

[4] Examples of analyzed data can be: statistics of source and destination of messages, their length, frequency, and other characteristics. Particular messages can be consequently remitted to cryptographic analysis based on these statistics and/or on knowledge of inherent properties/shortages of used protocol and algorithm.

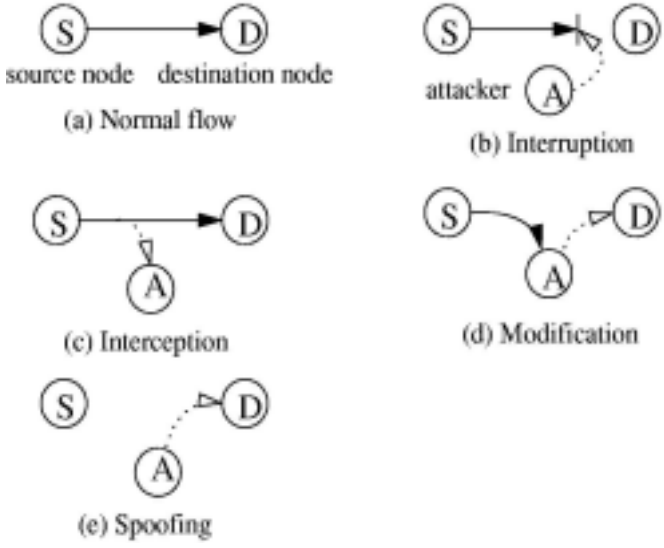


Fig. 1. Active attacks: interruption, interception, modification and spoofing

Basically, these attacks can be divided to:

- **masquerading** which means faking identity,
- **replay attack** based on capturing messages and their consecutive retransmission in order to avoid authentication and/or authorization,
- **modification** based on modification of captured messages in order to unauthorized pass authentication or to provide the addressee with non true information,
- **denial of service** preventing or inhibiting communication facilities, and
- **corruption of integrity and confidentiality** which is mainly related to mobile agents; it is based, e.g., on changing executive code of agents, etc.

The intentions of above stated attacks can be summarized to:

- **gain unauthorized access** to access/modify confidential/trusted information and/or to take control over facility and its resources,
- **impersonate another agent or modification** some information either to gain secrets or to deface the attacked agent, etc., and
- **prevent or prohibit communication facilities** and to deface them.

The most important of the above mentioned attacks are depicted in Figure 1.

Some kinds of attacks concerning capturing and consequent retransmission of messages with or without modification of the message are really very difficult to be detected as the captured messages may be replayed within their valid time window, especially when original messages are suppressed. This is for example the case of the reply attack.

3 Security Services Overview

The following points give a short overview of possible security services (countermeasures against potential attacks). The division is informative only as in fact many services are very closely interrelated and they can overlap:

1. **Confidentiality** protects communication and data from passive attacks as eavesdropping, traffic analysis, and disclosure. Confidentiality is applicable to the whole data stream or to a specific field in a message.
2. **Authentication** allows agents to prove their identity each other, i.e. to verify whether the counterpart is what it claims to be. The authentication service should not only serve as a validator of identity but along with other services should assure that no intruder agent will masquerade as one of the legitimate participant in the course of the communication (session).
3. **Authorization and access control** allows to limit and to control access of agents to particular resources; the resource requesting agent is authenticated first and then it is verified if it is granted to access the requested resource. The resulting operation over the requested resource is also dependent on the type of requested/granted access – it can be, e.g., read-only, write-only, etc.
4. **Transaction fairness** enables to force responsibility to agents for their behavior, to force them to fulfill their obligations, etc., simply, to keep mutual confidence within the agent community. Transaction fairness is required mainly in e-commerce and in open environments where some entities act on behalf of others. As an example, the non-repudiation can serve; it means that a sender of a message can not deny that he/she is the originator of that message.
5. **Integrity** means that messages are not duplicated, modified, reordered, replayed, etc, i.e. they are safe. Concerning mobile agents it means that agents are not deleted or modified.
6. **Accounting and auditing** service keeps tracks of events within the system and enables consecutive analysis, provides evidence of taken actions, commitments of agents, etc. Besides, this service is required by the transaction fairness service.

Hierarchy of services and their dependencies is depicted in Figure 2. General requirements for all above listed services can be summarized as follows:

1. **security** which means resistance against all potential attacks,
2. **reliability and bearing availability and fault tolerance** which means that services are accessible all the time despite possible hostile environment and/or agents, intruders, etc. By the fault tolerance it is meant that services are able to recover from some failures as disruption of the connection, lost of synchronization, etc.
3. **transparency and operability** ensuring both minimal effort of agents and minimal knowledge of underlying mechanisms necessary to utilization these services, and
4. **scalability** which means independence on number of agents, services, etc., and easy maintenance of the whole system.

A general countermeasure against attacks summarizes Table 1. This table is intended to be informative only as some countermeasures overlaps and some are efficient under special conditions, not in general.

As general mechanisms against masquerading, reply attack, and modification, except those stated in Table 1 sequence numbers,[5] timestamps,[6] and challenge-response protocols[7] can be mentioned [14]. These methods are often combined.

There is no general and utter countermeasure against the denial of service attack. In addition, a potential solution is more or less impossible without interaction with low level protocols. Some prevention can be based on analysis of recent communication within the system recorded by the accounting service.

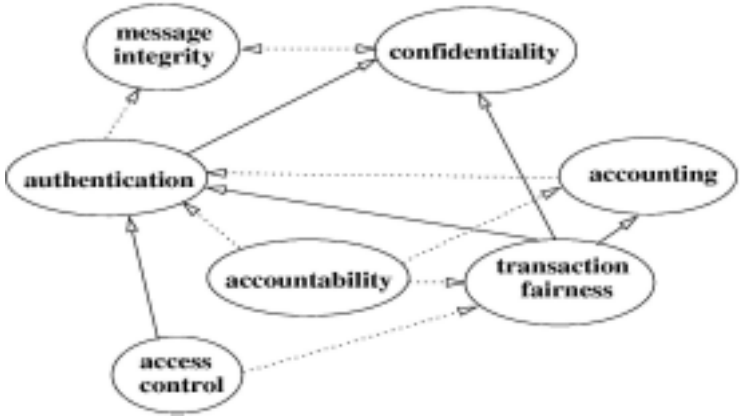


Fig. 2. Mutual service dependencies

Table 1. Attacks and their countermeasures

	confidentiality	authentication	authorization	integrity
eavesdropping	yes			
masquerading	yes	yes	yes	
reply attack	yes	yes	yes	yes
modification	yes			yes
denial of service				
integrity corruption	yes	yes	yes	yes

[5] Every message has attached a number, e.g., serial number, and it is accepted by its addressee if the number is that expected. The main drawback is the necessity to keep a track of last sequence numbers and in the case that the message stream is disrupted, i.e. the synchronization is lost, a new sequencing has to be established.

[6] Each message, key, certificate, etc. has a timestamp defining a time window indicating time span when the entity is valid. Obvious problems concern the proper size of the window; it must be large enough in order the message could be transported to its addressee and small enough to prevent reply attacks and clock synchronization. Another apparent problem is the necessity to relatively accurately synchronize clocks over network.

[7] The originator sends a challenge (nonce) to addressee to be authenticated and expects consecutive response - the sent value mostly modified by in advance agreed function. If the response is that expected the originator believes the identity of the addressee. Nonces are usually encrypted by the shared secret key. This is, in the connection with time window, the most used method as it is clock synchronization independent.

4 Additional Secure Architecture Requirements

In addition to services described in paragraph 3 we introduce following extra requirements for the proposed system:

- **Openness**, i.e., it should be possible to extend existing system with new services/mechanisms.
- **FIPA compliance**; the system must remain FIPA compliant even with supported security mechanisms. It concerns speech acts, services, etc.
- **Vendor or patent algorithms independence** which means that we do not prescribe any particular algorithm; instead we introduce general protocols enabling to handshake an arbitrary algorithm/protocol for particular sessions.
- **Single login access** to the system[8] which means that an agent, within its domain (see later), need not to prove its identity every time it wants to take some secure operation but just once, e.g., during startup.
- **Session oriented communication**; this requirement flows from the previous two requirements; it means that if an agent wants to communicate with another in a secure manner, it has to handshake protocols, algorithms and other conditions first. Agents can cache these sessions (identified by unique identifier) for future use.[9] The initial handshaking will be supported even by directory facilitators which will keep information about agent's security capabilities.[10]
- **Domains**; it can be assumed that, in reality, an operator, e.g. an enterprise, will run more than one agent platforms which should obey the same security policy (e.g., within an enterprise). By a security policy we understand an uniform maintenance of all involved agents, their secret keys, memberships in groups, associations of roles, rights for accessing some services, and the trust management.
- **Roles and groups** are useful for authorization purposes; they makes maintenance and function of the authorization service easier and more effective as instead of authorization of particular agents, groups, and/or roles are authorized.[11] We suppose roles and groups to be unique within a given domain (and all related sub-domains); all groups and roles will be automatically propagated to sub-domains. Any agent can be assigned to arbitrary number of roles and groups.
- **Legacy systems** – we require that our security services will be capable to work even with legacy systems, using their proprietary security mechanisms, with minimal required changes.
- **Periodical refreshment of all secrets** within the system. That means that all keys, certificates, vouchers, etc. will have limited their lifetime. This prevents disclosure and replay attacks.

---

[8] It is planed the single login will achieved together with the mandatory registration [4] to the AMS during the agent's birth to the agent platform.

[9] This idea is already used, e.g., in the SSL.

[10] The session oriented communication is designed to security purposes, the original FIPA message oriented communication (within established sessions) will retain. The lifetime of a session will be limited.

[11] In most cases we expect that the number of agents is much bigger than number of roles and groups.

5 Design Decisions

It seems reasonable to suppose an agent platform to be either secure or insecure, while the decision about platform security and security policy is up to the platform owner, see [13].

- **Distinction of inter- vs. intra- communication cases.** We believe that there are so different requirements to inter- vs. intra- communication that it is reasonable to treat those two cases independently. An arbitrary communication on Internet is the example of the inter-communication case, communication within an enterprise is the example of intra-communication. It is apparent that the complexity of secure communication on Internet will be more expensive and complex particularly when all mechanisms are to be open and universal. Even requirements in both cases will be slightly different.
- All authenticated and/or authorized communication will be encrypted with shared secret keys (symmetric encryption), so called session keys. These keys will be agreed by communication parties during the authentication phase.
- Preference of selection of crypto-algorithms for authentication will be dependent on the previous decision – for inter-platform communication we will provide trusted platform mechanisms (only for agents belonging to the same domain) and public-key cryptography. On the other hand, all intra-platform security will be provided with symmetric key encryption, i.e. identity of an agent will be proved by knowledge of shared secret key.
- For authentication with public keys we decided to use protocol designed by Woo and Lam[12] [12]:

Algorithm 1. Public key authentication algorithm

- $A \rightarrow AMS_A$: ID_A, ID_B
- $AMS_A \rightarrow A$: $E_{KPub\ A}[ID_B, KPubB]$
- $A \rightarrow B$: $E_{KPubB}[N_A, ID_A]$
- $B \rightarrow AMS_B$: ID_B, ID_A, $E_{KPub\ AMS\text{-}B}[N_A]$
- $AMS_B \rightarrow B$: $E_{KPriv\ AMS\text{-}B}[ID_A, KPub_A]$, $E_{KPubB}[E_{KPriv\ AMS\text{-}B}[N_A, K_{A\text{-}B}, ID_A, ID_B]]$
- $B \rightarrow A$: $E_{KPubA}[E_{KPriv\ AMS\text{-}B}[[N_A, K_{A\text{-}B}, ID_A, ID_B]], N_B]$
- $A \rightarrow_T B$, $A \rightarrow B$: $E_{K\ A\text{-}B}[N_B]$
- $B \rightarrow_T A$

[12] $A \rightarrow B$: M means that A sends message M to B, $E_K[M]$ means encryption of the message M with key K, where KPubA is a public key of agent A, KPrivB is a private key of agent A, KA-B means a shared session key used for symmetric encryption, KB means which AMS shared secret key used for authentication purposes, and $A \rightarrow_T B$ means that A trusts that B proved its identity. N stands for a nonce, T for a timestamp, AMS for Agent Management System.

- See Figure 3, case (1). The session key K_{A-B} is generated from nonce N_A to be fresh.
- For authentication within an agent platform based on trust management we decided to use protocol proposed by Neuman and Stubblebine [12], see Figure 3, case (2a)[12]:

Algorithm 2. Shared secret key authentication**.**

- $A \rightarrow B$: ID_A, N_A
- $B \rightarrow AMS$: ID_B, N_B, E_{KB} [ID_A, N_A, T_B]
- $AMS \rightarrow A$: E_{KA} [ID_B, N_A, K_{A-B}, T_B], E_{KB} [ID_A, K_{A-B}, T_B], N_B
- $A \rightarrow_T B$, $A \rightarrow B$: E_{KB} [ID_A, K_{A-B}, T_B], E_{KA-B} [N_B]
- $B \rightarrow_T A$

- Authentication across agent platforms but within the same domain will be accomplished by the similar algorithm as the intra-platform authentication, the only modification consists in forwarding the authentication request from AMS_B to AMS_A, see Figure 3, case (2b). Such the forwarding is useful if the addressee trusts the sender and vice versa. Trusts will be maintained by the domain trust management system and domain hierarchy.
- Every authentication will be reciprocal mandatory, i.e. if agent A wants agent B to be authenticated, it must authenticate to agent B as well.
- Message integrity will be provided by message authenticator computed by a hash function; the hash will be further encrypted with HMAC algorithm [7].
- Message compression will be provided on demand. The reason for introducing compression is both the length of common FIPA messages and the embarrassing decoding of messages by a potential eavesdropper.
- Maintenance of public keys and all shared secret keys, for authentication purposes, will be managed by the agent management system (AMS) facilitator. This augmentation of responsibility is natural as every agent born on a given platform has to register itself with the platform's AMS; during this mandatory registration agents will be authenticated as well. This bears on the single login approach requested in the above stated requirements.
- For authorization purposes we decided for the push model with centrally maintained privileges [2, 14]. In short, it means that the agent which requests a service has to first acquire a valid "access granting voucher" (central approach) and in virtue of this voucher the service providing agent will grant access to the requested resource (push model), in fact both decisions are very closely related[13]. This approach is similar to that used in Kerberos [6] or Sesame [11].

[13] The discussion about reasons which made us for that solution is beyond extend of this paper but simply speaking the two main arguments for the selected approach were: ease of maintenance of the whole system and tendency to minimize agent's load related to the security which bears with our requirement to maximal transparency and operability. All approaches are in detail described in [14].

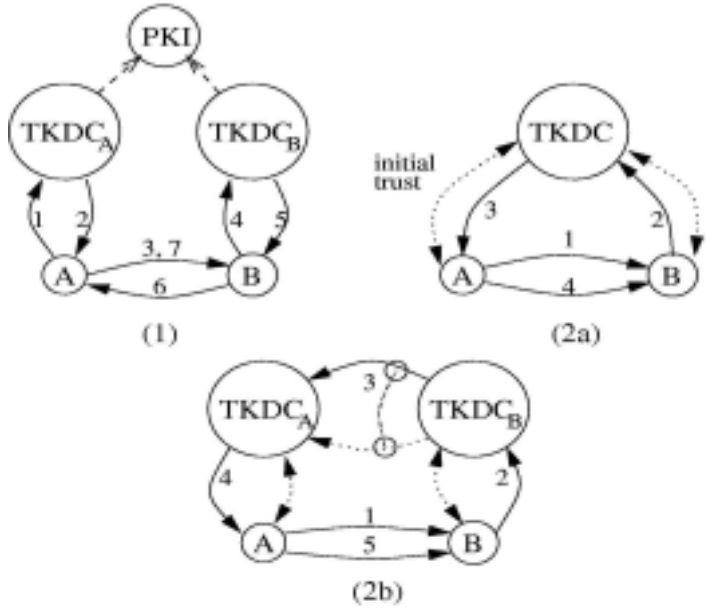


Fig. 3. Figure 3: Authentication cases. TKDC (Trusted Key Distribution Center) will be in our case the Agent Management System (AMS)

For issuing access granting vouchers and maintenance of assignment of roles and groups to particular agents the directory facilitator (DF) will be responsible. Augmenting its responsibility in this sense is again natural as the DF according to the FIPA specification [4] serves as yellow pages. In practice it means that the DF will held information about services and their access lists[14] (ACL) {<agent, service, ACL>}, information about agents and their assignment to roles and groups {<agent, role/group>} and information about assignment of ACLs and groups and roles {<ACL, role/group>}. In addition, for authentication of the DF, which was selected for registration of services by an agent, to that agent a special shared secret key will be generated during the first registration. The whole process can be described by the following algorithm:

Algorithm 3. Services registration and authorization**.**

- B → DF: authenticate and register available services
- DF → B: authenticate and send confirmation and generated shared secret key
- A → DF: authenticate and send request for the service and type of access
- DF → B: access granting voucher for A and authentication of A
- B → A: authenticate and send access granting voucher for A together with the requested service

[14] Access lists will be context based which means for their validity some condition(s) must be fulfilled. A context may be represented, e.g., by time of day, location of the agent, etc.

For the future we plan to extend capabilities of the access granting voucher about possibility of forwarding that voucher to another agent. Such the extension will enable to act agents on behalf of others. Rights of the forwarded voucher will be possible to limit compared to the owner of the voucher. In conjunction with the extended trust management and other services it will enable even to "trade" with such the vouchers.

Directory facilitator (DF) will serve even as the domain controller.

The problem with legacy system will be solved by access rights mapping at the level of particular agents.

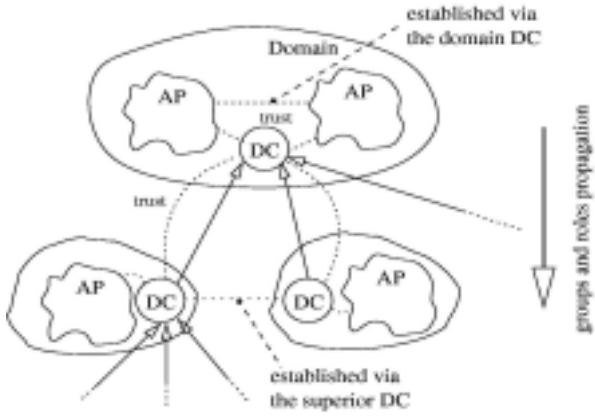


Fig. 4. Domain hierarchy and its trust management.

Except above stated requirements for domains we decided to extend functionality of domains about the trust management (see notes). In addition we assume it would be convenient to enable administrative splitting of domains to arbitrary number of sub-domains. Such hierarchically settled domains will constitute a tree – Figure 4. Each domain will be controlled by a special agent called domain controller (DC); due to the hierarchical settlement every domain controller (except that in the root of the tree – called the main domain controller) will have its superior controller which will be responsible for propagation of groups and roles (unique within the whole tree), the trust management, etc. Names of domains will be also hierarchical; they will be maintained by domain controllers as well.[15] As domains are very closely tight to authentication and authorization and they will be maintained by directory facilitators we decided to make directory facilitators responsible even for the domain controller functionality.[16]

[15] They will be hierarchical in the same way as names on Internet are. E.g., research.cvut.cz will have sub-domain agents.research.cvut.cz, etc.. The main domain controller would be root.cvut.cz.

[16] It may seem that we burden almost all responsibility for the security (trust management, issuing of access granting vouchers), and policy management (name and trust management) to directory facilitators. We must realize that both all those services are closely tight and in

The main sense for introducing trust to the security system is significant reduction of necessary authentications, especially in the case of secret shared keys. In our case the trust management will be administered by domain controllers, see Figure 4, by connections among a domain controller and all its sub-domains – if the operator set that some of two sub-domains have to be trusted (trust each other) the domain controller will generate and send to both of them a special secret key which enables them to authenticate each other. The key will be accepted by them as there is an implicit and in advance established trust between them and this superior domain controller. It can be proved easily that this solution will significantly reduce the number of shared secret keys within domains – from $O(N^2)$ to $O(\log N)$.

The trust is silently used even in public-private key systems, e.g., when an entity asks for a public key from scratch.

We introduce trust as a binary relation Agent x Agent → {true, false} expressing that the first agent trusts the second agent that it is what it claims to be. We will denote this relation $\rightarrow_T$. So introduced relation has, in general, two non-trivial qualities – it is not symmetric, i.e. A $\rightarrow_T$ B is not equivalent to B $\rightarrow_T$ A, and it is not transitive, i.e., A $\rightarrow_T$ B & B $\rightarrow_T$ C does not imply that A $\rightarrow_T$ C. Consequences of these qualities to authentication, authorization, and trust forwarding enabling agents to act on behalf of others will be published in our paper being prepared.

6 Conclusions

Basic requirements, presumptions, and effort to the achievement of FIPA agent security, which are currently being implemented within our multi-agent infrastructure, are discussed in the paper. To be honest, there are additional requirements concerning the FIPA standard augmentation to support all features of the proposed architecture – namely introduction of public-key infrastructure (PKI) interconnection, accounting agent, trust forwarding, etc. - these are not discussed and exceeded the limited extend of the paper.

Acknowledgments. This work was partially supported by the Ministry of Education of the Czech Republic under the Project LN00B096.

References

1. Ashley, P.: Authorization for a large heterogeneous multi-domain system. In Australian Unix and Open Systems Group National Conference, 1997.
2. Ashley, P, Broom, B.: Survey of secure multi-domain distributed architectures. Technical Report FIT-TR-97-08, FIT, 1997.

---

an agent platform there can be more than one directory facilitator so that the load can be equally balanced among more directory facilitators.

3. Homburg, P. C., The architecture of a Worldwide distributed system. PhD thesis, Vrije Universiteit, 2001.
4. http://www.fipa.org.
5. Jansen, W. A.: Mobile agents and security. Special publication 800–19, Institute of standards and Technology, 1999.
6. Kerberos – http://web.mit.edu/kerberos.
7. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-hashing for message authentication, 1997. RFC2104.
8. Krintz, C.: Security in agent-based computing environment using existing tools: A survey.
9. Neuman B. C.: Proxy-based authorization and accounting for distributed systems. In Proceedings of the 1993 PSRG workshop on network and distributed system security. 1993.
10. Schneier, B.: Applied cryptography. John Willey & Sons, 1996.
11. SESAME – http://www.cosic.esat.kuleuven.ac.be/sesame.
12. Stallings, W.: Cryptography and network security. Principles and practice. Prentice Hall, 1999.
13. Vlcek, T., Zach, J.: Considerations on secure FIPA compliant agent architecture. In. Proc. of IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS '02). V.Marik, L.M.Camarinha-Matos, H.Afsarmanesh (Eds.). Kluwer Academic Publishers, Boston/Dordrecht/London, pp. 11–124, 2002.
14. Vlcek, T., Zach, J.: A survey of security approaches in distributed and multi-agent systems. Technical report CC-2002, CertiCon a.s., 2002.
15. Wobber, E., Abadi, M., Burrows, M., Lampson, B.: Authentication in the TAOS operating system. SRC research report 117, Digital systems research center, 1993.
16. Wong, H. C., Sycara, K.: Adding security and trust to multi-agent systems. In Proceedings of autonomous agents '99 Workshop on deception, fraud, and trust in agent societies, 1999.
17. Wulf, W. A., Wang, C., Kienzle, D.: A new model of security for distributed systems. CS technical report CS-95-34, University of Virginia, 1995.

Agent Exchange – Virtual Trading Environment

Jiří Hodík, Milan Rollo, Petr Novák, and Michal Pěchouček

Gerstner Laboratory, Department of Cybernetics
Czech Technical University in Prague
Technická 2, 166 27 Prague, Czech Republic
{hodik,rollo,novakpe,pechouc}@labe.felk.cvut.cz

Abstract. Agent Exchange is a virtual trading environment serving as a test bed for experiments with market simulations, trading strategies and auctioning techniques. Agent Exchange is the distributed environment implementing up-to-date knowledge from multi-agent systems and secure communication. The developed agent community consists of independent agents communicating via defined protocols and ontology. The secure agent communication is an important part of this project. Roles that agents can play in the Agent Exchange community are Trading-agent, Bank-agent, Exchange-agent, Scenario-agent, User-agent, and Central treading authority agent. This paper describes the design of the Agent Exchange project, its functions and implementation details.

1 Introduction

Agent Exchange (AX) is a test-bed for experimenting with advanced trading strategies in an Agentcities [1] distributed environment. By developing the publicly available agent-based virtual market we aim to stimulate the research in the area of distributed decision making for B2B and B2C, and increase exploitation of the Agentcities environment for other than tourism-like services. The developed AX environment serves for experiments in electronic markets and as a tool during courses at artificial intelligence that are given by Gerstner Laboratory, Czech Technical University in Prague.

Provided infrastructure consists of six agent roles. Five of them (Trading-agent, Bank-agent, Exchange-agent, Scenario-agent and User-agent) form business community and participate on trading. The minimal number of agents acting any role is one and maximum number is unlimited for any agent role in the business community. The sixth one (Central treading authority agent) supports the other agents by broadcasting information about existence of community members. There is only one instance of this agent in the AX at this moment. Each agent has unique certificate including agent's role, and keys for encrypting or signing messages. The developed protocols and ontology allow start of the community and its negotiation. Traders can control the accounts in banks, get information from exchanges, and put the bid to sell/buy to the exchange. Exchanges can check traders' solvency and realize commodity transactions.

The AX environment applies security mechanisms of encryption and signing provided by X-Security [2] package. Signing and encrypting of whole and part

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 179–188, 2003.

of message are used in the communication. The implemented auction model is Continuous Double Auction. The defined communication protocols and ontology are the only stuff that must be supported and utilized by the agents acting in AX. Developers can the run pre-prepared agent community and are also encouraged to design and implement new ones. Both of adaptation and substitution of existing agents are allowed in the AX virtual market environment.

This paper contains the following parts. Section 2 aims to definition of seller and buyer, and their negotiation. Section 3 describes social model and their advantages for agents acting on virtual markets. Section 4 aims new developed virtual trading environment, its components and agent communication details.

2 Market Negotiation

The trade is usually based on the exchanging of commodities and services. The one who produces any commodity is the producer of this commodity. Equally the consumer is the one that consumes commodity. In this description we assume one of the exchanged commodity being universal currency, e.g. money. Having defined money we can describe seller as the one who owns commodity different to money and wants to exchange this commodity to money. On the other hand we are allowed to define buyer that has money and wants commodity. The seller wants to gain as much money as possible and buyer wants to pay as little as necessary to gain the commodity.

The seller and buyer are roles on the market and anyone is allowed to act both roles in the same moment and resell bought commodity to others. Beside seller and buyer there is a coordinator (manager) organizing the market negotiation. Coordinator role can be played by anyone playing seller or buyer role, or by another one subject, which is not interested in buying or selling commodity. This subject provides services of coordinating the trade for buyers and sellers.

All sellers and buyers have set limit value of the commodity. Each seller has minimum value he is compliant to get when selling one piece as well as each buyer is compliant to pay for one piece only some maximum value. These is a *private value* of a trader. In case of disclosure of this information other traders can modify their prices offered to the one with tapped limit value. The result of disclosure is reduction of a profit of handicapped trader. Beside the private values of particular traders there is the *common value*. Common value is a result of the negotiation mechanisms of traders and describes how the market valuates the commodity. Taking common value into account traders can increase their profit (e.g. sellers can asks higher price then his minimum one is if he assume to get it - commodity is sold and profit increased).

Various mechanisms exist for finding a trade partner and accomplishing the commodity and money exchange. To select a mechanism one must assume that the trade is one-shot or repeat. Realizing the repeated long-term contract one can get better conditions (e.g. price) from the seller or buyer than conditions got if the trade negotiation is started every time from null. Even in long-time

contract it could be possible to get temporary worse condition (e.g. pay more) then actual one-shot market offers. It takes place if the contract provide better conditions over whole period of the contract than finishing it because of temporary conditions. For one-shot sale and purchase it is more effective to find seller or buyer offering the actual best price. The last thing that trader must realize is whether he knows the common value of the offered/required commodity. Suitable mechanisms can reduce the lost and enlarge the profit. The main mechanisms are Contract Net Protocol (CNP) and auctions. CNP was originally described by Smith in [3]. It is based on calling others for proposals and selection of the offer that best fits the requirements. Auctions collect offers of interested sellers and buyers. Different auction mechanisms require different bidding and can reach different results.

Auctions are supposed to be significant in the future of electronic trading because auction is setting the price of goods according to the actual demand and supply. Reynolds in [4] defines auction as a competitive method of allocation limited commodity. There are more then one types of auction and not all of them are based on ascending prices.

Reynolds [4], Sandholm [5] and others (e.g. [6]) divide auction into 4 basic types depending on sealed or open bids and ascending or descending price. As another auction type they consider Double Auction. Almost all various auctions used in the world can be described as variants of the 4 basic and Double Auction. Unfortunately this system of classification is not consolidated in names of types. We use the same names for auction types like [4] and [5] do. The basic 4 auctions are:

- English (ascending-price),
- Dutch (descending-price),
- First-Price sealed-bid (first-price),
- Vickrey (uniform second-price).

Another auction type is Double Auction widely used in the variant called Continuous Double Auction (CDA). We decided to use CDA as a basic auction type for Agent Exchange. Other types of auctioning mechanisms are also envisaged to be implemented.

3 Social Models

As mentioned before the trade is based on the commodity exchange. The conversion rate (ratio of common values of exchanged commodities) mainly depends on market demand and supply of particular commodities. The real price that seller can get and buyer pays also depends on their knowledge about the market and other traders. Not all traders know all the others, not all traders negotiate about the price and try to pay as least as possible, and not all traders can wait to expected seasonal price-cutting. Having knowledge about other traders' desires, their private values, negotiation abilities, and seasonal trends can help

trader considering this knowledge to increase the profit. Not only knowledge about competitors but also about teammates sharing the same goal can improve bargaining abilities.

The work [7] defines the concept of a social knowledge. Social knowledge in a trading environment describes the others' behaviors, offered and required commodity, amount of commodity and private values. Teammates provide all information accurately because they contribute to maximizing the profit of the team. On the other the information the competitors announce is only the one they suppose to increase their own profits. Information that the trader never says to others is his *private knowledge*; the information given to the trade partners (e.g. during market negotiation) is the *semiprivate knowledge*; and the information accessible to everyone (e.g. trade's communication address) is the *public knowledge*. To get information about the others one can ask them or use subscribe/advertise protocol if others are compliant to provide required information. Everything else about other traders must be estimated from their announced requests, responses to requests for offers in contract net protocol, and their bidding on auction if bids are not sealed. In multi-agent systems there is an acquaintance model serving as the tool for knowledge representation. Typical example of modern acquaintance model is the tri-base model [8].

4 Agent Exchange

AX business community roles (See figure 1) are Trading-agent, Bank-agent, Exchange-agent, User-agents and Scenario-agent. Another agent is Central trading authority agent that does not participate on trading but is an important member in open communities.

4.1 Agent Roles

- **Trading-agent** sells and buys commodities. Its aim is to make profit. Trading-agents are allowed to cooperate.
- **User-agent** is an interface between a human user and the Trading-agents. This agent manages the team of Trading-agents that a human user owns. Each team has its own team-leader.
- **Bank-agent** organizes payments for the inter-agent trading. The Bank-agent administers the money and commodity accounts for the agents that ask Bank-agent to do it.
- **Exchange-agent** is responsible for organizing the trades among the Trading-agents. This agent is envisaged as an open environment for implementing various auctioning strategies.
- **Scenario-agent** influences the trading environment by shipping and delivering specific commodities into/from the market.
- **Central trading authority agent** informs Trading-agents about existing Bank-agents and Exchange-agents. The other function of this agent is providing money and commodity to newborn Trading-agents to allow them deal on the market.

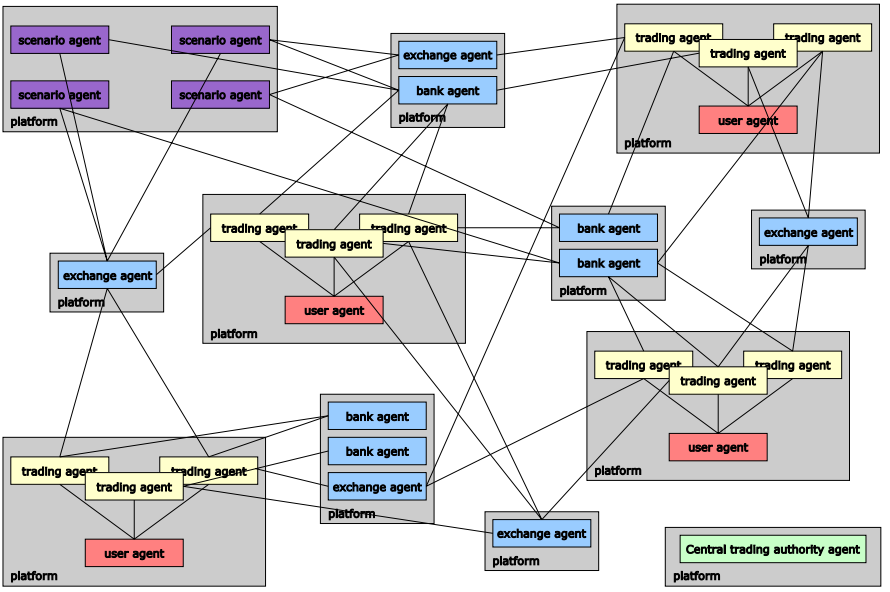


Fig. 1. Agent Exchange community structure

Trading-agent (TA). The Trading-agent is an agent that exploits its resources and rationality in order to make profit on transforming different types of resources by indirect interaction with the other Trading-agents. The interaction of Trading-agents is made available by the Exchange-agent. Trading-agent operates according the orders from User-agent. The easiest task for the Trading-agent is to exchange specified commodity on specified auction. More difficult task is to exchange defined amount and price of the good but the selection of the auction is up to Trading-agent. The most difficult task is to cooperate in trading with the teammates. It is most difficult but by the cooperation the team can get higher profit. Cooperation of Trading-agents and the User-agent is described in the following section.

User-agent (UA). The User-agent serves as an interface between team of Trading-agents and a human user. The task of this agent is to manage the Trading-agents that a particular user owns and to collect and show up-to-date information about accounts and Trading-agents' states. The team of the User-agent and subordinate Trading-agents tries to reach the profit and satisfy the human user owning the team. The team is required to contain intelligence allowing trading and some graphic user interface (GUI) to present state of the team to user.

The easiest way how to create team of a User-agent and Trading-agents is to create one agent compiling the features of both required agent types. This new agent can be viewed as one-member team with implemented graphic user interface. It does not need any social model of the teammates because of nonexistence of them. The social model that this agent and also agents using other variants of cooperation needs is the model of competitors and Exchange-agents and their exchanges on that Trading-agent trade. This variant is required in some simulations of lone self-oriented investors without sharing resources. This model is only one that is implemented in the AX project now.

For the further use by another developers we described protocols and ontology used by Trading-agent to allow developers implement their own Trading-agents and User-agents. Using these protocols and ontology developers are able to implement an interface between the Agent Exchange community and their expert and decision making systems. They do not need to implement their existing software again to prove it in the Agent Exchange.

Bank-agent (BA). The Bank-agent is a registering agent that organizes payments for the inter-agent trading. The Bank-agent administers accounts for subscribed agents. Besides money there are other commodities registered with the Bank-agent. The Bank-agents serve as a virtual commodity storage. Thus Bank-agent contains databases of agents' accounts. Each record contains account number, identification and communication address of owner, commodity that is stored on this account, state (number of items or money), and a list of bookings. The booking is record about amount of commodity that account owner trusted the Exchange-agent to manipulate with it. All the Bank-agents are required to act trustworthily. Contemporary Bank-agent cannot ask charges for its services. The feature of service charges will be implemented if necessary in any simulation.

Exchange-agent (EA). The Exchange-agent receives offers to buy or sell from Trading-agents. All bids must contain offered commodity, requested commodity, price, deadline till the offer is valid, and commodity or money booking certificate. The Exchange-agent checks all necessary bank accounts whether the seller owns enough commodity and buyer enough money. When conditions to satisfy newcome offers can be accomplished the exchange is done by bank orders to exchange the money and commodity. Then the seller and buyer are informed about the executed transaction.

As a basic auction model we implemented Continuous Double Auction (CDA). This one is also used in another virtual trading environments such as TAC [9]. In the real financial world CDA is used in stock exchanges (e.g. NYSE New York Stock Exchange [10]). The CDA is the auction that never stops and sellers and buyers place they required and offered prices independently. At the moment when the requirements of any seller and buyer meet, the exchange of the commodity and money is executed.

We decided to allow to exchange only money to commodity and commodity to money on the auction. Barter is not allowed now and because the real trading is based on money. We plan to use it only if simulation explicitly requires it. Although barter is not used in the AX now it can be allowed anytime when a simulation requires it. Amount of sold and bought commodity is allowed to be bigger then 0. Commodity unit and virtual money unit are divisible.

Scenario-agent (SA). The Scenario-agent is the first producer and final consumer of all commodities in the AX market environment. Scenario-agent sells and buys the commodity according to orders from Power User (human user, who controls the commodity flow to/from the Agent Exchange environment). The Scenario-agent acts in causal manners that could be informally described and publicly known in a possible combination with a random factor. Scenario-agents act directly on exchanges. To control Scenario-agent Power User can use predefined scenario file describing scenarios of periodic affecting or graphic user interface for immediate affecting. Each record in the scenario file contains these data:

- **Probability** of performing the action,
- **Time** of action that defines start-up of the action,
- **Type** of action defines if the commodity has to be sold or bought,
- Offered or requested **commodity**,
- Commodity **amount**,
- Commodity **price**,
- **Exchange** which will be asked to provide the action.

The loss making and ability to make money and commodities according the orders is the reason why Bank-agents are required to receive commodity not only from the other Bank-agents as usual in the ordinary transactions. They have to receive commodity from Scenario-agents too to allow them create products for selling. Thus the Scenario-agent is allowed to generate commodity for influencing the market.

The procedure of influence is as follow. According to orders read from the file, or received from the Power User via GUI, Scenario-agent orders Bank-agent via Central trading authority agent to create both of source account and target account if it is necessary. Next step is generating commodities that have to be sold or money that have to be spent. Then the Scenario-agent sends relevant requests to exchanges. Scenario-agent requests them to buy and sell commodities at the amount defined in scenario file or by Power User. When matching bids are found the Exchange-agent performs the trade via standard protocols for commodities exchange.

4.2 Communication Ontology

We use three ontologies for messages construction: Management-ontology, Bank-ontology and Exchange-ontology. Management-ontology provides messages controlling the community. Bank-ontology is the main one for business communication and serves all messages that other agents send to (and receive from) the

Bank Agent. Trading Agents communicate to the Exchange Agent via Exchange-ontology. Bank-ontology and Exchange-ontology form group of inter-agent business communication ontology.

4.3 Communication Security

The business communication messages are confidential and often contain the orders to transfer money and other commodities. To secure the communication within Agent Exchange we use the X-Security model [2]. This model provides partial security for community members and is being tested on the Agent Exchange community.

X-Security model supports securing of whole message content by signing and encrypting as well as securing a defined content part. When the complete content is secured the message is extended with new slot called X-Security. This slot contains identification of security method and used keys. When the content is signed this slot also contain the signature. The information included to X-Security slot is necessary for the receiver of the message to choose appropriate method for decryption or signature check. When only part of message content is secured the message content contains X-Security information beside the secured part.

Signing specified part of message serves as an authorization of another agent to perform some action. The authorized agent can copy the authorization and send it to agent that has to perform this action, which originally could be requested only by authorizing agent. The agent performing the requested action has a glue that the action was really requested by agent who can do it or this agent dedicated another one to request it.

Authorizing another agents in AX is necessary because exchanging is provided by sending orders to Exchange-agents. The Exchange-agent receives bids and relevant authorization, which are signed by Trading-agents, for commodities that have to be sold. After finding corresponding bids The Exchange-agent orders Bank-agents, where commodities the goods and money to be exchanged are stored, to transfer them to the buyer's and seller's accounts. Bank-agents can do it only after previously receiving of correct authorizations that account owners provide to the Exchange-agent. This one passes them to the Bank-agents to gain access to the commodities and to give Bank-agents proofs that the commodity owners really requested transfers of the commodity and money.

The basic communication security functions for AX business community are provided by Security certification authority agent (SCA). Each agent is required to register its security certificate at SCA to be allowed to use secure communication. The security certificate contains agent's identification, time of certificate validity, agent's security level, public keys, keys identification and type (e.g. RSA/1024 or SHA with DSA/1024). SCA maintains actual valid certificates of all agents and provides them to every agent by request.

When any agent receives a message and the agent does not have appropriate certificate for its decryption or signature-check, the agent asks SCA for it. After receiving the certificate the agent can decrypt message or check message signature and continue in communication with the original message's sender.

4.4 Implementation and Future Work

Agent Exchange is the new publicly available virtual multi-agent test-bed. The aim of our research is to provide the virtual electronic market environment to developers who want to test and prove their algorithms for real electronic trading. For implementation we use Java 1.4 and Jade 2.61. The AX community contains agents of six different roles. These agent roles allow implementing almost any subject participating on trading in real business world.

The future work that has to be done is an improvement of Trading-agent strategies to allow them cooperate and extending the Bank-agent by the service charges collecting. Next step of our research we see in interconnection of the Trading-agent and the Scenario-agent to allow any agent transform commodities.

5 Conclusion

E-business and virtual marketplaces become significant in these days. The AX project aims to develop an agent-based virtual market infrastructure that can be used as a test-bed for electronic software trading agents before their run in the real competitive environment. The developed system does not contain any central system that is necessary for trading. Any agent participating on the commodity exchange is substitutable and can be removed and also added to the market anytime during the run of the simulation. Publication of used protocols and ontology allows developers to design new agents (acting defined roles) and after implementing the interface connect and test existing system in virtual market environment. As an example of multi-agent system the Agent Exchange is used by students attending courses at artificial intelligence that are given by Gerstner Laboratory, Czech Technical University in Prague.

Source codes of Agent Exchange project are available at the project official web-page [11].

Acknowledgement. We thanks for funding the Agent Exchange project by Agentcities.NET (IST-2000-28384) as a project ACNET.02.28.

References

1. Agentcities.NET. *Official web-page*. http://www.agentcities.net.
2. P. Novák, M. Rollo, J. Hodík, T. Vlèek. Communication Security in Multi-Agent Systems, In: V. Marik, J. Müller, M. Pìchouèek (editors), *Multi-Agent Systems and Aplications III*. Springer-Verlag, 2003.
3. R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.
4. K. Reynolds. *series of articles about auctions for Agorics, Inc.*, http://www.agorics.com/Library/auctions.html.
5. T. Sandholm. Distributed Rational Decision Making. In: Weiss, G. (ed.), *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*. MIT press, 1999.
6. Auction server XSPIPE.*Official web-page*. http://www.xspipe.com/help/auctiontypes.cfm.
7. M. Pěchouček, V. Mařík, and J. Bárta. A Knowledge-Based Approach to Coalition Formation., In: *IEEE Intelligent Systems*, (3):17–25, 2002.
8. M. Pěchouček. V. Mařík. and O. Štěpánková. Role of Acquaintance Models in Agent-Based Production Planning Systems, In: M. Klusch L. Kerschberg (editors), *Cooperative Information Agents IV* – LNAI No. 1860, Eidelberg, 2000.
9. TAC 2002 server. *Official web-page*. http://www.sics.se/tac/.
10. NYSE New York Stock Exchange. *Official web-page*. http://www.nyse.com.
11. Agent Exchange. *Official web-page*. http://agents.felk.cvut.cz/agentexchange/.

Adding OWL Semantics to Ontologies Used in Multi-agent Systems for Manufacturing

Marek Obitko and Vladimír Mařík

Gerstner Laboratory, Department of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University, Prague, Czech Republic
{obitko,marik}@labe.felk.cvut.cz

Abstract. Ontologies are needed for agent communication as they provide means for describing intended semantics of the language used for expressing the content of messages. We show what types of ontologies and their descriptions are used in the area of multi-agent systems for manufacturing and compare them with the ontology modeling languages in the semantic web area. We present a framework that enables adding semantics of OWL ontologies used in the semantic web to the ontologies without any formal explicit description used in the manufacturing domain (expressed usually in XML). The ontologies with explicitly defined semantics can be used for purposes that require formal semantics such as for automated reasoning over the source ontologies from the manufacturing domain or for integration of manufacturing and semantic web agents. This integration is one of the key issues for successful development of agent-based solutions for manufacturing as well as virtual enterprises.

1 Introduction

Communication in multi-agent systems is possible only in the case that the communicating agents share their ontologies [6]. In other words, agents have to understand the messages that they receive from other agents. Even when a message is delivered over the network and syntactically understood (i.e. an agent is able to recognize syntactical elements), there is one more step needed — the agent has to understand the meaning of words and constructs used in the message content.

The actual meaning of the message content is captured in a message ontology. The possibilities of ontology description range from a simple vocabulary to a full logical theory. The choice of ontology description then determines how much a prospective ontology sharing or reuse will be possible. A better formal description often means improved possibilities for automatic processing.

In this paper, we are particularly interested in ontologies that are currently used in multi-agent systems for manufacturing [16]. As mentioned later in the paper, these ontologies are explicitly described usually only informally. By informal description we mean a description that is not directly processable by computer agents, such as a general natural language description. Languages for better description of ontologies that capture better the intended semantics are

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 189–200, 2003.

being developed in the semantic web area [8]. We take inspiration from the semantic web ontology languages and propose methods for adding more semantics to the ontologies used in multi-agent systems for manufacturing nowadays. We present a proof-of-concept framework that enables to add semantics together with the mapping from the original ontology to the new one. We build on our previous work [12] where we explained the needs for adding semantics to ontologies as well as the need for integrating both the manufacturing and semantic web agents.

Adding explicit formal semantics to ontologies without any explicit formal description enables to use automated reasoning for ontology sharing and integration. The integration of ontologies used in the area of MAS for manufacturing with those developed for the semantic web is one of the key issues for successful development of many practical systems.

The paper is organized as follows: First we describe ontologies in general and then briefly compare ontologies used in the manufacturing area with those used in the semantic web field. From the semantic web area we focus on the Web Ontology Language (OWL) that is supposed to become World Wide Web Consortium (W3C) recommendation for the semantic web. We give some reasons for using OWL for ontology modeling. We show how the ontologies used in the manufacturing area are insufficiently specified and thus we describe ways that can be used for adding semantics to them. A proof-of-concept framework for adding semantics to the ontologies in the manufacturing domain will be presented as well. The work with the framework is illustrated on a simple example from the transportation domain [16]. We conclude with a very brief discussion of how the formal ontology can be used by an agent for reasoning purposes.

2 Ontologies in Agent Communication

The ontologies specify the conceptualization [6] of the domain. When agents want to communicate, they have to share the ontology used for communication. This means that they share the conceptualization of the domain, i.e. that they primarily recognize the same objects and the same relations among the objects in the domain they operate on. Also, they have to share the specification of this conceptualization, i.e. they have to use the same language and the same words for describing the same objects.

The ontologies capture the structure of the domain, i.e. they define how to model the state of affairs in a domain together with possible restrictions. They capture knowledge that is not changing (or that is changing very rarely), while a particular knowledge base or a particular message captures a particular state in the domain concerned.

The ontologies should be well designed and well defined. By a good design we mean that they should adequately capture the modeled domain for the task at hand. By a good definition we mean not only the syntax, but also the semantics. The formal semantics is important if we want to use an automated reasoning over ontologies, such as automatic ways of ontology integration and sharing of different ontologies. To enable the sharing of ontologies, ontologies must be explicitly described in a way understandable to agents.

3 Ontologies Used in Manufacturing Domain

The design of multi-agent systems for manufacturing is influenced mainly by the Foundation for Intelligent Physical Agents (FIPA) [3] standards developed to ensure agents' compatibility at various levels. FIPA uses Open Knowledge Base Connectivity (OKBC) as a base for expressing ontologies [4]. The advantage of this approach is that a direct mapping from ontologies to an object-oriented language is possible. For example, in the JADE package for construction of the FIPA-compliant multi-agent systems, the recommended way of handling ontologies is to create Java classes describing ontology classes and relations and to register them as a part of the application ontology. This is a practical, fast way of creating an ontology with an immediate underlying implementation.

A typical example of ontology specification used in this domain is the FIPA Ontology specification itself [4], where the specification is provided in a tabular form. The tables contain name of a term and its informal description in a natural language. It is possible to use the predicates of FIPA-META-Ontology ontology to specify the content of the ontology. These predicates are intended to provide a way to exchange information about ontologies among agents, but they are not used for a formal description of ontologies.

As also noted in [12], we can see that the ontologies currently used in the manufacturing domain are driven mainly by the need for having something that would work as soon as possible. The attention is focused on the task that a particular multi-agent system should solve, not on the interoperability with other agents from other communities. It is not expected that agents will need to be able to communicate with agents from other communities, so no formal ontology description is available.

These ontologies have their semantics explicitly described at best in a form of a textual description of the intended use [4,16]. The description in the form of textual description does not fulfill the requirement of having an explicit description understandable to agents. The semantics is described formally in agent's code, however this is an implicit description that cannot be used to reason over ontologies. No explicit formal description of ontology is available. The need for a formal ontology description is arising, for example, when an easier automated integration with other agents, either with manufacturing agents that were not designed to work together, or with other agents like semantic web agents is needed. A typical example involves different agents from different companies that were originally not designed to work together in one supply chain [12].

4 Semantic Web and Web Ontology Language (OWL)

The specification of ontologies as described in the previous section is in contrast with the ontologies used in the semantic web area. The semantic web [8] is a vision of extension of the current web in which the information would be given well-defined meaning that would enable better processing by computers. In addition to the current pages, machine understandable information in a form of a formally defined content with formally defined ontologies for that content should be provided in order to enable more intelligent automated processing. As

it is necessary to enable semantic web agents to handle more ontologies and to process them automatically, a formal specification of ontologies is needed.

There have been many proposals for languages that could be used for these purposes. In this paper, we focus on the Web Ontology Language (OWL) [7] that is intended to provide a language that can be used to describe ontologies (together with knowledge bases) in a form of classes and relations among them together with further restrictions and intended use of them. It is designed primarily for the WWW documents and applications, but it can be used for any other domain as well. OWL is a language intended for knowledge and ontology representation — it is not simply a language for a message or data format, as is for example XML. OWL was designed by W3C based on the experience gained with the DAML+OIL [1] language. OWL is intended to replace DAML+OIL. Like DAML+OIL, OWL is built on RDF triples [10], and uses RDF/RDFS and XML Schema [15] constructs.

4.1 Description of OWL

In OWL, the ontology is a set of definitions of classes, properties, and constraints on the way those classes and properties can be employed. The OWL ontology may include the following elements [7]:

- taxonomic relations between classes,
- datatype properties (descriptions of attributes of elements of classes),
- objects properties (descriptions of relations between elements of classes),
- instances of classes and properties.

There are three increasingly complex species of OWL. OWL Lite provides the simplest constructs that allow for classification hierarchies and simple constraint features. More powerful OWL DL includes the complete OWL. It is a language that is in correspondence with the OWL description logic, that allows automated reasoning. The most complex OWL Full includes also the complete OWL, and in addition to OWL DL, it provides the freedom of RDF. For example, it is possible to use individuals for a class definition (i.e. to define a class by its extension). We will consider the OWL Full (i.e. the language with no restrictions) in this paper.

OWL divides the universe into two disjoint parts. One part is the datatype domain described by the XML Schema [15] datatypes. The other part is the object domain that is described by the OWL classes.

All the class elements in OWL create a subclass of `owl:Class` (a subclass of `rdfs:Class`). Two class names are predefined — `owl:Thing` and `owl:Nothing`. Every object is a member of `owl:Thing` and no object is a member of `owl:Nothing`. Classes can be refined by elements that include:

- `rdfs:subClassOf` asserting that a class is a subclass of a class expression (which includes a simple class, see below)
- `owl:disjointWith` and `owl:sameClassAs` asserting that a class is disjoint with or equivalent to a class expression

- boolean combinations of class expressions — `owl:intersectionOf` asserting conjunction, `owl:unionOf` asserting disjunction, and `owl:complementOf` analogous to logical negation, but restricted on objects only
- objects properties (descriptions of relations between elements of classes)

The class expression is a class name, enumeration, property restriction on a class, or a boolean combination of class expressions.

The other part of an ontology definition in OWL is the definition of properties of classes. Properties can be either object properties (instances of `owl:ObjectProperty`) that relate objects to other objects, or datatype properties (instances of `owl:DatatypeProperty`) that relate objects to datatype values. Datatype values are defined by XML Schema definitions. Similarly as in the definition of classes, a property can be refined by the following elements:

- `rdfs:subPropertyOf` asserting that it is a subproperty of another property
- `rdfs:domain` asserting that the property only applies to instances of the specified class expression
- `rdfs:range` asserting that the values of the property are only instances of the specified class expression
- `owl:samePropertyAs` asserting equivalence to other property
- `owl:inverseOf`, `owl:TransitiveProperty`, `owl:SymetricProperty`, `owl:FunctionalProperty` and `owl:InverseFunctionalProperty` asserting additional properties of properties

We can now return to property restrictions on classes. These restrictions are a special kind of class expressions. They refine the classes by restricting possible values of properties of these classes, but do not restrict properties themselves. Restrictions on properties for classes can be described using the following elements:

- `owl:allValuesFrom` that is analogous to the universal quantifier of the predicate logic and that defines the class of all objects for which the values of the property belong to the class expression. A similar construct, `owl:someValuesFrom`, is analogous to the existential quantifier
- `owl:hasValue` that defines the class of all objects for which the property has at least one value equal to the object or datatype value
- `owl:cardinality`, `owl:maxCardinality`, `owl:minCardinality` that all restrict the cardinality of the property, i.e. how many distinct values of the given property is allowed for a class.

4.2 Reasons to Use OWL

The Web Ontology Language (OWL) that is going to be standardized by W3C provides an ontology modeling language with defined formal semantics. XML provides standard common syntax for parsing documents or messages. OWL provides a standard way of expressing the semantics. XML itself does not provide any explicit description of intended use of data. The semantics is expressed only implicitly in the agent code and is thus bound to a specific agent. There are no

means how to reuse the information about semantics for other agents. With this approach, each agent has the semantics hard wired which leads to the necessity of programming the understanding for each agent separately, may cause different interpretation in different agents, and makes dynamic discovery and exploitation impossible.

Expressing the semantics explicitly has advantages of separating the semantics from the program code and thus leads to writing less code for implementing mutual understanding for multiple agents. It also reduces the chances of misinterpretation in various implementations. OWL provides a common language to define semantics so that anyone can understand it. OWL uses the XML syntax and builds on RDF syntax and semantics, i.e. it builds on existing widely used technologies so that common XML and RDF tools can work with OWL as well.

OWL is going to become the base ontology modeling language for the semantic web. There are already tools available both for authoring and parsing various forms of OWL documents as well as tools for reasoning over ontologies expressed in OWL. An example of the usage of the OWL specified ontology is provided in the section 8.

5 Semantics of XML Ontologies Used in Manufacturing Domain

From the description of the OWL compared to the form of ontologies currently typically used in the manufacturing domain, it is obvious that the manufacturing ontologies lack explicit formal description in a form understandable to computer agents. The syntax is usually described in a sufficient way — for example by the DTD or XML Schema or at least by the ontology classes definition in a particular implementation language. However, the semantics is at best expressed only informally by a natural language description.

For specifying XML intended use, Document Type Definition (DTD) and XML Schema Definition (XSD) are used today. These languages are capable to express a structure of a document. However, the need for adding ontology modeling capabilities to XML for better specification of intended use has been already recognized and several proposals are available. These capabilities allow representing semantics, not only the syntax of a document as pure XML and DTD does.

These proposals can be divided into two groups. The first group contains approaches that extend the XML definition (DTD or XSD) with extensions that allow adding more semantics to XML. These approaches include the Schema for Object Oriented XML (SOX) [2], XML++ [9]. They both allow using constructs that are available in the object-oriented world, such as inheritance or polymorphism. However, the target is a proprietary language that is not widely used. Another example can be found in [5] where a RDF-like constructs are introduced into XML. The other group includes approaches that translate XML to other languages (such as RDF) using a fixed translation algorithm [11] or that provide formal semantics [14] based on some assumptions of the XML use.

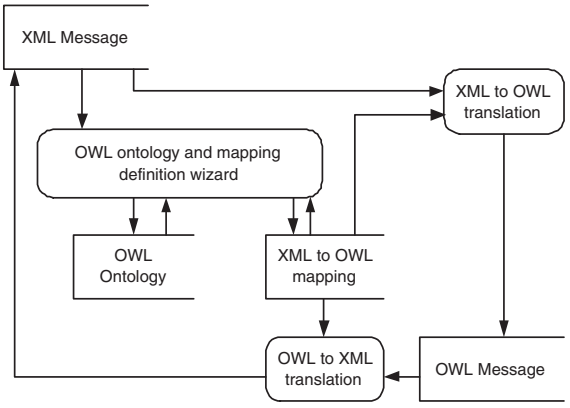


Fig. 1. Data flow diagram of translating XML to OWL with added semantics. The ontology and mapping definition wizards enable to define the mapping that is used for translation between OWL and XML.

6 Adding OWL Semantics to XML

We present an alternative approach to the proposals described in the previous section. Our intention is to provide semantics in a form of OWL constructs and define the semantics by providing a mapping from XML to OWL.

Instead of using proprietary extension, we use the emerging standard for the semantic web, the Web Ontology Language. Instead of enabling a fixed translation, we provide more alternatives of translation. However, a human assistance is required to select the alternative of intended semantics of given XML constructs. Only XML documents or messages are required, no other definition is necessary.

The data flow diagram for our approach is shown in Figure 1. We present an OWL ontology and a mapping definition wizard that enables to define the OWL ontology and mapping between XML and OWL. Using this information, messages can be translated between OWL and XML, either for communication between manufacturing and semantic web agents, or for using better description of XML messages and ontologies for enabling automated reasoning over them.

The intended application scenario is as follows. XML messages are taken as the input for the wizard that helps the user to determine the target OWL ontology and mapping between the original XML message and the target OWL ontology. Both the ontology and the mapping are continuously refined with new messages. The resulting mapping can be then used for translating messages in both directions between the original XML form and the OWL form with the specified ontology.

6.1 Example from the Transportation Domain

Let us describe the processes and data stores used in the data flow diagram in Figure 1 on a sample XML message content from the transportation domain

[16] shown in Figure 2. The semantics of the XML elements in the message is typically described as a brief natural language explanation of meaning.

```
<component type="ConveyorBelt" name="b1">
   <connection from="w1" to="d1" defaultCost="12.5">
</component>
```

Fig. 2. Example XML message content from the transportation domain [16].

This message is an input for the ontology and mapping definition wizard. The wizard parses the message and suggests how to construct the OWL ontology from it and how to translate from the XML content to the OWL content. The target OWL content is based on the defined OWL ontology. Several heuristics are provided that guide the translation. A default choice is always provided, but the user can change the choice if necessary. This process has two outputs. The first result is the OWL ontology describing the target ontology for translation. This ontology can be further modified for better description of the semantics. The other result is the mapping between the source XML format and the target OWL format that is based on the designed OWL ontology.

The OWL ontology is described in the standard OWL, so that any OWL enabled agent or tool can work with it. The mapping information is used for translation from the XML message to the OWL message or in the opposite direction. The full specification of the message in OWL is then the conjunction of the OWL message and the OWL ontology that defines resources used in the message.

6.2 Heuristics Used for Translation

There are several heuristics that are used by the wizard for determining the mapping and the target ontology. We will briefly describe them now. The wizard handles the XML elements, attributes, and texts included in the elements. Texts are handled as special attributes of XML elements. Since there can be more texts within an element, the wizard enables to attach a different OWL construct to each of them. The correspondence is determined by their order, as ordering of all the constructs is important in XML (in opposite to OWL, where the order is not important). XML can be thought of as an ordered tree, while OWL is a general directed graph.

The supported possibilities of mapping from the XML elements and attributes with their relations expressed in the XML document into OWL constructs are the following ones. First of all, the XML element can be mapped into the OWL class, and the XML attribute can be mapped into the OWL datatype property. The names of the OWL classes or attributes can be copied from the names used in XML. The attribute may also refer to an object in a domain, so it is also possible to map an attribute to an object property with a particular class having another datatype property. Another possibility is to map a pair of an XML element and its attribute with particular value to a single OWL class. This is useful when the attribute indicates further specialization of the element

```
<transport:ConveyorBelt> a <owl:Class> ;
   rdfs:label "ConveyorBelt" ;
   rdfs:subClassOf <transport:TransportationEdge> .
<transport:targetNode> a <owl:ObjectProperty> ;
   rdfs:label "targetNode" ;
   rdf:type <owl:FunctionalProperty> ;
   rdfs:subPropertyOf <transport:connectedTo> ;
   rdfs:range <transport:TransportationNode> ;
   rdfs:domain <transport:TransportationEdge> .
<transport:defaultCost> a <owl:DatatypeProperty> ;
   rdfs:label "defaultCost" ;
   rdf:type <owl:FunctionalProperty>
   rdfs:range [ a <xsd:real> ] ;
   rdfs:domain <transport:TransportationEdge> .
```

Fig. 3. Fragment of the extended OWL ontology (in the N3 notation that is shorter than the XML serialization) definition for the XML message from the Figure 2.

in the intended model. Embedded elements can be handled in a similar way as attributes — they can be mapped to classes that are connected by an object property with parents or they can be handled directly as datatype properties. Unlike with attributes, the object property is the default choice here. Another possibility is to handle attributes of embedded elements directly as properties of a class corresponding to the parent element.

```
<transport:ConveyorBelt rdf:ID="#b1">
     <transport:name>
          <xsd:string xsd:value="b1"/>
     </transport:name>
     <transport:sourceNode rdf:resource="#d1"/>
     <transport:targetNode rdf:resource="#w1"/>
     <transport:defaultCost>
          <xsd:real xsd:value="12.5"/>
     </transport:defaultCost>
</transport:ConveyorBelt>
```

Fig. 4. OWL message (in XML serialization) that was translated from the XML message in Figure 2. Resources `d1` and `w1` are of type `TransportationNode` defined from other parts of the original XML ontology. The namespace transport refers to the ontology in Figure 3. Note that the description is more understandable than the original XML form — there is an individual `b1` of type `ConveyorBelt` (which is the subclass of `TransportationEdge` according to the ontology), and it is connected to its source and target nodes (that are individuals defined elsewhere) by its properties. Also, it is clear which individual the `defaultCost` property refers to.

The wizard parses the XML message, and offers the user to choose from options based on the described heuristics or to accept the default pre-selected choice. After the user selects from the offered heuristics, both the OWL ontology and the mapping are updated. Figure 3 shows a fragment of the OWL ontology

definition that is based on a result of accepting the choice selected in the wizard. Using the mapping information, the sample message from Figure 2 is translated into the OWL form that is shown in Figure 4. Note that the resulting message together with its ontology is more clearly specified than the original XML message (see the note in the description of Figure 4) and represents more natural specification of domain conceptualization and the particular state concerned.

The resulting OWL ontology can be later enhanced by additional information that would provide more semantics using any OWL enabled tool. In this way, the target OWL ontology may be very different from the original XML ontology structure, but it is still possible to translate between these two formats. The only requirement is to preserve classes and properties generated by the wizard.

6.3 Implementation

The proof-of-concept framework is implemented as a standalone application in Java using the Xerces2 XML parser[1] and Jena Semantic Web Toolkit[2] for processing information based on RDF. The described wizard takes XML fragments as input and allows the user to specify their meaning in OWL and thus also add the semantics to the original XML fragments. The framework also allows translating the XML fragments to OWL based on the mapping rules selected by the user. The library for translating messages from XML to OWL based on the ontology and mapping defined in the framework can be integrated into multi-agent system for manufacturing. Such a system needs either to work with better-defined semantics for e.g. interoperability purposes, or needs to communicate with OWL-enabled semantic web agents. Either a new dedicated agent that would translate messages from XML to OWL and back can be introduced or each agent can get this additional functionality provided by the library.

7 Usage of OWL Explicit Ontology Specification

Let us very briefly illustrate what an agent can derive from the knowledge expressed in the ontology specified in Figure 3. The `owl:FunctionalProperty` type specifies that both `targetNode` and `defaultCost` properties are functional properties, i.e. that there exists at most one value for that property. Note that this does not say anything about specifying the value in a particular document or message — as DTD or XML schema would do. It refers to the overall knowledge about the domain (the value can be specified elsewhere and may be even inaccessible for a particular agent). Let us suppose that there are several different kinds of identifications of a particular transportation node — one is by its purchase identification number from a commercial department, another one is the address used between transportation agents, and another one is its serial number from the transportation node manufacturer. When an agent gets information from several sources that these three transportation nodes are target nodes of one transportation node, it can immediately derive from the functional

[1] `http://xml.apache.org/xerces2-j/`
[2] `http://www.hpl.hp.com/semweb/`

property type that these three nodes are the same individuals that just have different identifications for different communities.

In the specification in Figure 3 the `targetNode` is `rdfs:subPropertyOf` of `connectedTo`. If the `connectedTo` property is declared to be `owl:SymetricProperty`, and an agent knows that a particular transportation node is a target node for a particular transportation edge, it can derive not only that the node is connected to the edge, but also that the edge is connected to the node. Another example is usage of `owl:TransitiveProperty` for e.g. accessibility property in deriving what other nodes are accessible from a particular node.

Other examples include reasoning over taxonomies of classes as objects. OWL provides many primitives for refining the details of taxonomies of classes. An important usage of the ontological knowledge is automated establishing of mapping between ontologies for translation of information between different ontologies [13].

All of the described information is hard wired in the agent that uses the XML form. However, the formal ontological knowledge is expressed only implicitly and its usage is limited to what was preprogrammed in the agent code. With OWL, the ontological knowledge is available for reasoning to any agent.

8 Future Work and Conclusion

We have presented a workflow supported by a framework for adding the OWL semantics in relatively easy way. The approach does not require any special form of ontology and is not fixed in any way to provide the semantics. The workflow has been illustrated on a particular example from the transportation domain, but can be used also for any other ontology that needs to add the OWL semantics. The wizard is currently not very user friendly, since it may generate vast number of options if it gets a bit complicated message and does not support checking the selected options against the already existing OWL ontology. Also, the existing information about the OWL ontology and mapping can be used to heuristically guess additional OWL semantics and offer it to a user for confirmation. Future work includes improving the framework in this way and the integration of the translation engine into a multi-agent system. The need for adding semantics to the existing ontologies used in the manufacturing domain is arising, as different multi-agent systems are needed to be integrated together even when they were not originally designed to work together. Another reason for specifying the semantics formally is the integration with the semantic web and other agents to form agent-based virtual enterprises or to cross the borders between different companies. The presented approach with the proposed supporting framework enables easier integration of the current manufacturing agent-based solutions.

Acknowledgements. This work is supported by the Ministry of Education of the Czech Republic within the project No. 212300013. We would like to thank anonymous reviewers for their comments that helped us to improve the quality of this paper.

References

1. DAML – Darpa Agent Markup Language. http://www.daml.org/
2. Andrew Davidson et all. Schema for Object-Oriented XML 2.0 (W3C Note). 1999. http://www.w3.org/TR/NOTE-SOX/
3. FIPA – Foundation for Intelligent Physical Agents. http://www.fipa.org/
4. FIPA. Ontology Service Specification. 1998. http://www.fipa.org/specs/fipa00006/
5. Ramanathan V. Guha: Xemantics – Adding object Identity and Explicit Relations to XML. 2003. http://tap.stanford.edu/xemantics.html
6. Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition. Vol 5. 1993
7. Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn A. Stein: Web Ontology Language (OWL) Reference. 2003. http://www.w3.org/TR/owl-ref/
8. Jim Hendler: Agents and the Semantic Web. *IEEE Intelligent Systems.* Vol 16, no. 2, March/April 2001. pp 30–37.
9. Hasan Jamil, Giovanni Modica: An Object-Oriented Extension of XML for Autonomous Web Applications. *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management.* McLean, VA, USA. 2002.
10. Ora Lassila, Ralph R. Swick (eds). Resource Description Framework (RDF) Model and Syntax Specification (W3C Recommendation). 1999. http://www.w3.org/TR/REC-rdf-syntax/
11. Sergey Melnik: Bridging the Gap between RDF and XML. 1999. http://www-db.stanford.edu/~melnik/rdf/fusion.html
12. Marek Obitko, Vladimír Mařík: Ontologies for Multi-Agent Systems in Manufacturing Domain. *HoloMAS – Industrial Applications of Holonic and Multi-Agent Systems (DEXA Workshop).* IEEE Computer Society Press, 2002, pp 597–602.
13. Marek Obitko, Vladimír Mařík: Mapping between Ontologies for Agent Communication. *Multi-Agent Systems and Applications III, LNAI 2691.* Springer-Verlag. 2003
14. Peter F. Patel-Schneider, Jerome Simeon: Building the Semantic Web on XML. *International Semantic Web Conference (ISWC).* Springer-Verlag. 2002.
15. Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn. XML Schema (W3C Recommendation). 2001. http://www.w3.org/TR/xmlschema-1/
16. Pavel Vrba, Václav Hrdonka. Material Handling Problem: FIPA Compliant Agent Implementation. *Multi-Agent Systems and Applications II, LNAI 2322.* Springer-Verlag. 2002. pp 268–279.

Complex Data Integration Based on a Multi-agent System

Omar Boussaid, Fadila Bentayeb, Amandine Duffoux, and Frederic Clerc

BDD - ERIC - Université Lumière Lyon 2,
Batiment L, 5 avenue Pierre-Mendès-France
69676 BRON Cedex – FRANCE
Fax: +33 4 78 77 23 75, Tel: +33(4 78 77 24 58
boussaid@univ-lyon2.fr, bentayeb@eric.univ-lyon2.fr
{amandine.duffoux,frederic.clerc}@etu.univ-lyon2.fr

Abstract. The expansion of the WWW and the growth of data sources lead to the proliferation of heterogeneous data (texts, images, videos, sounds and relational views). We call these data "complex data". In order to explore them, we need to carry out their integration into a unified format. Collecting, structuring and storing constitute the different tasks of complex data integration. There exists many approaches for data integration like mediated schemes and wrappers, or warehousing. In this paper, we propose a new approach for complex data integration that uses both classical warehousing approach and multi-agents systems (MAS) technology. We consider the different tasks of the data integration process as services offered by actors called agents. To validate this approach, we have implemented a multi-agent system for complex data integration named SMAIDoC. One of the advantage of The MAS technology is that it provides an evolutive structure to our system.

Keywords: data integration, complex data, services, agents, Multi-Agent System.

1 Introduction

The data warehouse and OLAP (On-Line Analytical Processing) techniques [1, 2] proved reliable in the field of data management, particularly for numerical data. In a data warehousing process, the phase of data integration is crucial. Many methods for data integration have been published in the literature.

The development of Internet increased the proliferation of different types of data (images, texts, sounds, videos, databases...) that we call complex data. Exploiting these data requires the use of powerful tools to facilitate their integration: data acquisition, structuring and storage, etc.

In this context, we propose a new method for complex data integration, based on a data warehousing approach associated with a Multi-Agent System (MAS). Our objective is to take advantage of the MAS that are a set of agents to perform

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 201–212, 2003.

the phase of complex data integration. Indeed, we consider the different tasks of the data integration process as services offered by agents.
Data extraction: This task is performed by the agent in charge to extract data characteristics from complex data. The obtained characteristics are then transmitted to the agent responsible for data structuring.
Data structuring: To perform this task, the responsible agent deals with the organization of the data according to a well-defined data model. Then, the model is transmitted to the agent responsible for data storage.
Data storage: This task is performed by the agent that feeds the database with the data by using the model supplied by the data structuring agent.

In order to validate our complex data integration method, we have implemented, SMAIDoC [3], a Multi-Agents System for Complex Data Integration. This system is a set of intelligent agents offering the various services that are necessary to the phase of integration of complex data. Moreover, allows to update the existing services and to add/create new agents. SMAIDoC is based on an evolutionary architecture that offers a great flexibility and a strong structuring.

This paper is organized as follows: section 2 presents a state of the art concerning the data integration approaches and agent technology. In section 3 we expose the issue of complex data integration and our approach. We explain in what the use of a MAS is adapted to carry out this approach and also present the architecture of SMAIDoC . Its implementation is described in section 5. Finally, we conclude this paper and we present research perspectives in section 6.

2 State of the Art

2.1 Data Integration

The nature of the web data is generally heterogeneous: relational or object databases, tagged documents or plain texts, images, sounds, videos ... The complexity and the proliferation of these data raise the problem of their integration. Several approaches have been proposed.

In the mediators based approach [4], the proposed solution maintains the data in their source and builds abstract views from which a mediator tries to satisfy the user's queries. The current architecture of this approach is based on a mediators-wrappers system [5]. It allows the query of distributed and heterogeneous data sources. It operates like a centralized and homogeneous system. The mediator performs the data integration by providing the user an homogeneous and global view of the system. Its task is to reformulate the user's queries according to the various contents of the accessible data sources. Several wrappers correspond to this mediator, one for each data source. Their role is to extract the data selected by the query according to the type of the corresponding data source. The major interest of this system is the query reformulation performed by the mediator, provides the user some flexibility for query writing and formulation. Furthermore, query approximation is integrated into the system to help the mediator to refine the user's query as well as the user always obtain an answer.

The data warehouse approach [1,2] consists in building a new database, called data warehouse, from various data sources. In this case, integration corresponds to the ETL process (Extracting, Transforming and Loading), which cleans and transforms the heterogeneous data before they are loaded in the data warehouse. The data warehouse is based on an analysis-oriented data model where data represent indicators (measures) that can be observed according to axes of analysis (dimensions). The model is multidimensional and characterizes a context of analysis. Queries are generally complex [6]. They need data treatments in order to summarize them and facilitate their interpretation. Moreover, they require sophisticated data navigation methods through various dimensions with OLAP operators [7]. The answers to the user's queries are represented as data cubes. The information on the indicators is aggregate and visualized according to different points of view [8].

2.2 The Multi-agent Systems

An agent software is a classical program that is qualified as "intelligent". Intelligent agents are used in many fields such as the networks (ANT [9]), on-board technologies (LEAP: Lightweight Extensible Agent Platform [10,11]) or human learning. An intelligent agent is supposed to have the following intrinsic characteristics: *intuitive* - it must be ready to take initiatives and to achieve the actions that are assigned to him; *reactive* - it must listen to the actions of its environment and acts in consequence; *sociable* - it must be able to communicate with other agents and/or users.

Moreover, agents may be mobile and can independently move through an acceptor network in order to perform various tasks [12]. A Multi-Agent System designates a collection of actors, called agents, that communicate with each other [13]. The purpose of this collection is to perform a specific task. Each agent is then able to offer specific services in an autonomous way and communicates the results to a receiving agent (human or software) and has a well defined goal . The MAS must respect the programming standards defined by the FIPA (Foundation for Intelligent Physical Agents) [14].

3 Complex Data Integration

The exploration of complex data needs a pre-processing phase in order to render data available to classical storage and analysis tools. There are several approaches associating the data warehouses technology with the Web. One of them is the *data webhousing* [15]. Other approaches hinge on distributed architectures [16].

In our approach, the main data source for a system offering complex data storage and exploitation services is the Web. We allow the user to target and to retrieve his data in order to build up a local data source. Data complexity is mainly due to type diversity. Data also enclose both structure and content information.

We have proposed in [17] a conceptual, logical and physical model for complex data. It consists in a whole process that helps describing the complex data and representing them with XML documents.

Complex data are defined as complex objects made up of one or several sub-documents. These sub-documents may have different types. The complex objects are represented by a UML class diagram (Fig 1). The UML model is a general model containing low-level descriptors as meta data [18]. It may be supplemented with a specific part containing semantic descriptors. All complex data collected on the Web or coming from any other data source may be represented using this conceptual model. Moreover, by using an XML DTD , the model is translated into an XML document that can be stored in a relational data base [19]. To

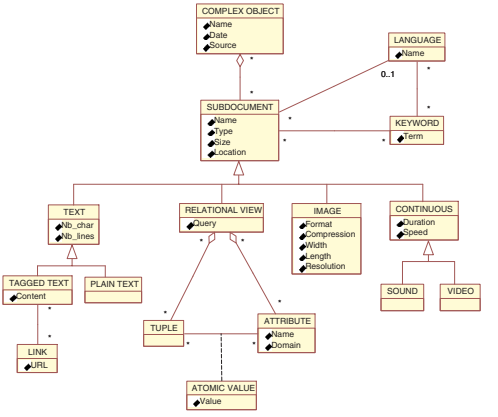


Fig. 1. Complex object conceptual model

validate this approach, we propose a MAS based system to integrate complex data in a local data source, which constitutes a first stage of the data warehousing process. The source code of this system we baptized SMAIDoC is available online [3]

4 SMAIDoC's Specifications

4.1 Motivation

Complex data integration is a succession of tasks . We can assimilate these tasks, which are not necessarily sequential, to services offered by well-defined actors in a system intended to achieve such an integration process. Therefore, it is natural to introduce the concept of agent. Agent technology provides a clear and modular structuring, an easy re-use and an ideal maintenance. Consequently, this

structuring allows to easily add services or agents to the system [20]. Furthermore, the agents communicate with each others without external intervention. The information exchanges are then simplified and the tasks'execution is made clear. Each agent performs a well defined task-set. Moreover, the concept of mobility is very important. An agent can move to an information source's location, process data and returns at its owner's location. It is able to independently create distant databases, associated to other agents, and to provide consequent services to the user. It is thus useful to collect relevant information from the different data sources such as the Web or local area networks.

4.2 SMAIDoC's Architecture

We present in this section the global architecture of the SMAIDoC system (Fig. 2), which is based on agents, services and products. Our system is based on a platform of generic agents. We have instantiated five *agents* offering *services,* allowing complex data integration. Each service generates *products.*

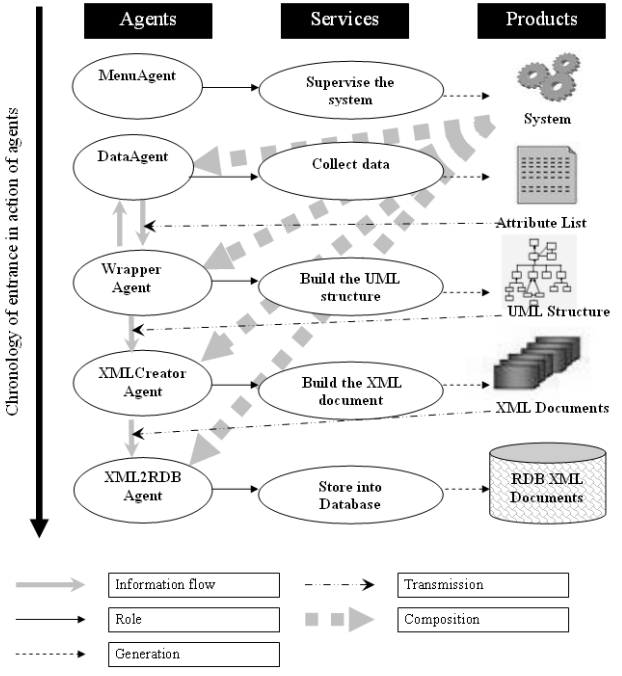


Fig. 2. SMAIDoC's architecture

Agents

MenuAgent pilots the system. It is the user interface. It supervises agent migrations and indexes the accessible sites from the platform.

DataAgent collects the data concerning the documents.

WrapperAgent instantiates the UML structure based on the data supplied by *DataAgent*.

XMLCreatorAgent builds valid XML documents from a DTD and the UML structure.

XML2RDBAgent stores XML documents in a relational database.

Services

All the tasks of the complex data integration process can performed in four steps.

Collecting: Build up a meta data collection from the documents containing the complex data. Meta data are obtained by feature extraction. The features are considered as attributes: e.g. text keywords, video sequence duration or image compression ratio.

Structuring: From the previously collected data, the presented model (Fig. 1) is instantiated. This operation generates a conceptual model describing the complex data.

Generating: From the instantiated model and by using an XML DTD, the complex data are expressed as valid XML documents.

Storing: The valid XML documents are mapped into a relational database.

Products

When the SMAIDoC system is activated, it initializes the environment (Fig. 3) where the agents evolve. The SMAIDoC agents operate on already collected complex data. Feature extraction is performed by an agent. It is used to instantiate the various classes of the UML model. The generated XML documents are used to supply a relational database which is the end-product provided by SMAIDoC.

4.3 Functional Structure of the Agents

In order to describe the structure and the behavior of a SMAIDoC agent, we propose a grammar expressed with a predicate-based formalism. Each agent has two components: the header and the body. The header identifies the agent (line 2) by its name and the name of the platform it belongs to (line 3). The agent's body specifies the possible scenarios: first of all, the agent is recorded into the platform (line 6) its behavior is defined (if it has a specific one) (line 7). Then, we define the the agent's mobility, if necessary, its migration (lines 9,10,11) and the inter-agent communication (lines 12,13,14). These actions are performed sequentially, in an iterative way.

1. < *Agent* >::=< *Agent_Header* >< *Agent_Body* >
2. < *Agent_Header* >::=< *Agent_ID* >
3. < *Agent_ID* >::=< *Agent_Name* >< *platform_Name* >
4. < *Agent_Body* >::=< *Agent_Configuration* >
5. < *Agent_Configuration* >::=< *Record* >< *Scenario_Adding* >
   < *Mobility* >< *Communication* >< *Agent_Configuration* >
6. < *Record* >::=Add < *Agent_ID* > in < *Platform_Register* >
   If it is not already made
7. < *Scenario_Adding* >::=Add < *Scenario* >to< *Agent_ID* > |*NoAction*
8. < *Scenario* >::= specific task autonomously accomplished
9. < *Mobility* >::=< *Mobility_Ontology* >< *Migrate* >
10. < *Mobility_Ontology* >::=Add < *Agent_ID* > to
    < *Plateform_Mobile_Agent_Register* >
11. < *Migrate* >::=Send < *Agent_ID* > to < *Host_Address* >
12. < *Communication* >::=< *Message_Receiving* > | < *Message_Sending* > |*Active_Waiting*
13. < *Message_Receiving* >::=< *Message_Waiting* >< *Message_Content_Reading* >
14. < *Message_Sending* >::= Send < *Message* >to< *Agent_ID_Destination* >
15. < *Message_Waiting* >::= Wait < *Message* >
16. < *Message_Content_Reading* >::= Read < *Message* >

4.4 SMAIDoC Agents Behavior

To illustrate the behavior of SMAIDoC agents, we consider a site S containing different complex data types.

Step 1: Until the last piece of complex data is read do: User selection of site S and migration command for agents *DataAgent* and *WrapperAgent*.

Step 2: *MenuAgent* orders these two agents to migrate. Migration is performed.

Step 3: Sequential collection of the meta data by *DataAgent* and instantiation of the UML structure by *WrapperAgent*. According to the data type, *DataAgent* retrieves the attributes and sequentially sends them to *WrapperAgent* for the gradual development of the UML structure.

Step 4: The structure is sent to the *XMLCreator* agent.

Step 5: *XMLCreator* builds the XML document from the UML structure and the associated DTD. It parses and stacks up the elements. According to the structure form, it compares the stacked elements with the structure components. It generates the corresponding XML code.

Step 6: Transmission of the XML document to *XML2RDBAgent*.

Step 7: *XML2RDBAgent* orders to map of the XML document in the database.

Step 8: Go to *Step 1*.

5 Implementation

To validate our complex data integration method, we implemented a prototype named SMAIDoC [3], a Multi-agent system for complex data integration, which is in charge to achieve the necessary different tasks required in the process of the complex data integration. Thus, we developed five JAVA classes that correspond to the agents defined in the SMAIDOc architecture (Fig.2).

The specificities of our application implied the use of the Java language [21]. Java is portable across agent programming platforms. The only restriction is

the presence of a virtual machine. We used JADE version 2.61 [22,23] which respects the standard defined by the FIPA. JADE proposes to build reception structures for agents: the containers. They are abstract elements located on any machine, but always on the same instance of the JADE platform. There are two default pilot agents. The *ASM* agent (Agent System Management) supervises the management of the agents present on the platform: their state, their site, their migrations, available sites of reception... The *RMA* agent (Remote Management Agent) provides an interface for JADE. Indeed, we have defined a new pilot, *MenuAgent*, which allows the user to select a site (among the available ones). This site contains complex data for integration. Once the site is selected, *MenuAgent* orders the migration of the agents *DataAgent* and *WrapperAgent* towards this site and the treatment can start.

The JADE agents communicate through a protocol called ACLMessage[1] [24]. The exchanged messages use an ontology in serializable format (text). In JADE, an ontology is represented as an automatically generated class further to the demand to send a message. This communication method is similar to the association of SOAP (Simple Object Access Protocol) and RDF that is described in [25]. In order to facilitate the system set-up, starting and initialization, we created a "launcher" which replaces the user for this tiresome task. The launcher creates the JADE platform and the virtual containers for agents reception. It also initializes the system pilot agents (RMA and DMS). Fig. 3 presents an overview of the system. The SMAIDoC system allows the user to select the doc-

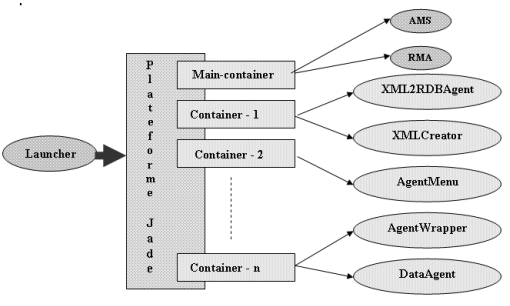


Fig. 3. Overview of the JADE platform

uments which will be integrated in a database. The following documents types are proposed: simple text, tagged text (HTML, XML...), image, sound, video and relational view. For each type, default attributes are proposed to the user: keywords, word or line count for a text, duration for a sound or a video, resolution for an image... We used three ways to extract the actual data: (1) manual

[1] to be standardized by the FIPA

capture by the user, through graphical interfaces; (2) use of the standard Java methods and packages; (3) use of ad-hoc automatic extraction algorithms. Our objective is to progressively reduce the number of manually-captured attributes and to add new semantic attributes that would be useful for later analysis and that could be obtained with, for example, data mining techniques.

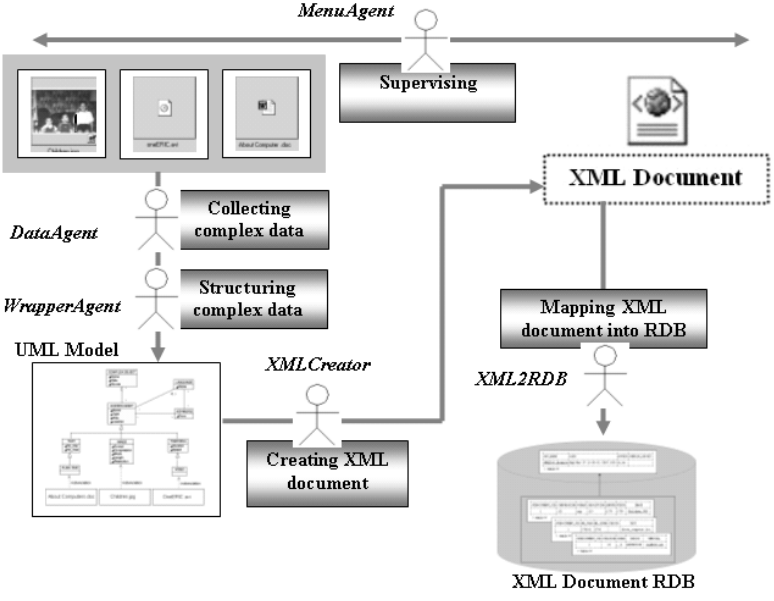
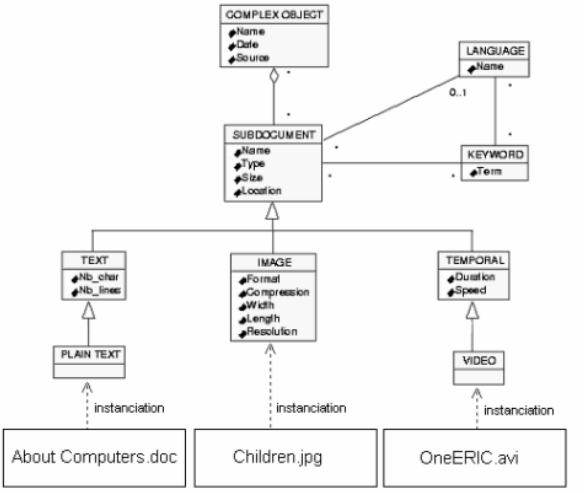


Fig. 4. SMAIDoC: Whole example of complex data integration process

The complex data collected from a given site are integrated into a database according to the process we described before. The various tasks are performed by the SMAIDoC's agents. They evolve in JADE containers defined on the same host machine. These containers can be implemented on different host machines. Therefore they generate an inter-machine mobility of the SMAIDoC agents. We show in Fig.4 how SMAIDoC integrates complex data from raw sources to relational databases. In this example, we only consider the complex object named *SMAIDoC_Example* composed of three documents that are image, video and plain text, represented by *Children.jpg*, *oneERIC.avi* and *AboutComputer.doc* files, respectively. Thanks to the different agents of SMAIDoC, these documents are collected, structured in UML class diagram (Fig. 1), translated into XML document (Fig. 1) and finally mapped into a relational database. We give in the following the relational model of our obtained database of XML documents.




```
<?xml versiobuffout; n =' 1.0'? >
<!DOCTYPE ComplexData SYSTEM 'c : /mlfd2.dtd' >
< COMPLEX_OBJECT >
< OBJ_NAME > Exemple_Smaidoc < /OBJ_NAME >
< DATE > Wed May 07 2003 10 : 08 : 02 < /DATE >
< SOURCE > files < /SOURCE >
< SUBDOCUMENT >
< DOC_NAME >About_Computer.doc< /DOC_NAME >
< TYPE > Texte < /TYPE >
< SIZE > 278016 < /SIZE >
< LOCATION > C : /About_Computer.doc < /LOCATION >
< LANGUAGE > English < /LANGUAGE >
< KEYWORD > Computer < /KEYWORD >
< KEYWORD > informatics < /KEYWORD >
< TEXT >
< NB_CHAR > 276281 < /NB_CHAR >
< NB_LINES > 1736 < /NB_LINES >
< PLAIN_TEXT > C : /About_Computer.doc
< /PLAIN_TEXT >
< /TEXT >
< /SUBDOCUMENT >
< SUBDOCUMENT >
< DOC_NAME > oneERIC.avi < /DOC_NAME >
< TYPE > Video < /TYPE >
< SIZE > 497152 < /SIZE >
< LOCATION > C : /oneERIC.avi < /LOCATION >
< LANGUAGE > fr < /LANGUAGE >
< KEYWORD > labo < /KEYWORD >
< KEYWORD > logo < /KEYWORD >
< TEMPORAL >
< DURATION > 20 < /DURATION >
< SPEED > 5 < /SPEED >
< VIDEO >
< THEME_V > ANIMATION < /THEME_V >
< /VIDEO >
< /TEMPORAL >
< /SUBDOCUMENT >
< SUBDOCUMENT >
< DOC_NAME > Children.jpg < /DOC_NAME >
< TYPE > Image < /TYPE >
< SIZE > 108191 < /SIZE >
< LOCATION > C : /ENFANTS3.JPG < /LOCATION >
< LANGUAGE > fr < /LANGUAGE >
< KEYWORD > family < /KEYWORD >
< KEYWORD > children < /KEYWORD >
< IMAGE >
COMPRESSION > 122 < /COMPRESSION >
< FORMAT > jpg < /FORMAT >
< RESOLUTION > 123 < /RESOLUTION >
< LENGTH > 1178 < /LENGTH >
< WIDTH > 1738 < /WIDTH >
< /IMAGE >
< /SUBDOCUMENT >
< /COMPLEX_OBJECT >
```

Fig. 5. SMAIDoC example : The sspecific UML Model and the XML document

1. Complexobject_db(OBJ_NAME, DATE, SOURCE, COMPLEX_OBJECT)
2. Temporal_rel(SUBDOCUMENT_PID, DURATION, SPEED, CHOICE, TEMPORAL)
3. Image_rel(SUBDOCUMENT_PID, COMPRESSION, FORMAT, RESOLUTION, LENGTH, WIDTH, IMAGE)
4. Text_rel(SUBDOCUMENT_PID, NB_CHAR, NB_LINES, CHOICE, TEXT)

The instanciation of the database with the complex object *SMAIDoC_Example* and its sub-documents is given in the following.

1. Complexobject_db(SMAIDoC_Example, Wed May 07 2003 10:08:02, files,)
2. Temporal_rel(1, 20, 5, ANIMATION, oneERIC.avi)
3. Image_rel(1, 122, jpg, 123, 1179, 1739, Children.jpg)
4. Text_rel(1, 276281, 1736, , About_computer.doc)

6 Conclusions and Future Work

In this paper, we proposed a new approach for complex data integration based on both data warehouse technology and multi-agent systems. Our proposed integration approach necessitates several tasks that must be assimilated to services offered by well-defined agents in a system intended to achieve such an integration.

We developed then, SMAIDoC, the system that allows this integration, based upon a flexible and evolutive architecture on which we can add, remove or modify services, and even create new agents.

Two agents named respectively, *DataAgent* and WrapperAgent are charged to model the input complex data into UML classes; the *XMLCreator* Agent translates UML classes into XML documents that are mapped in relational database with the *XML2RDB* Agent. Moreover, the different agents that compose our system are mobile and the services they propose coincide with the Extraction, Transformation (ETL) and Loading tasks of the data warehousing process.

We plan to extend the services offered by SMAIDoC, especially extracting data from their sources and analysing them. For example, the DataAgent agent can converse with online search engines and exploit their answers. In the other hand, we can create new agents in charge to model data in the multidimensional way and apply analysis methods like OLAP [8] or data mining techniques.

References

1. Inmon, W.: Building the DataWarehouse. John Wiley & Sons (1996)
2. Kimball, R.: The data warehouse toolkit. John Wiley & Sons (1996)
3. BDD-ERIC: SMAIDoC download page. http://bdd.univ-lyon2.fr/logiciels.php?id=4 (2003)
4. Rousset, M.: Knowledge representation for information integration. In: XIIIth Int. Symp. on Methodologies for Intelligent Systems (ISMIS 2002), Lyon, France. Volume 2366 of LNAI., Springer Verlag (2002) 1–3
5. Goasdoué, F., Lattès, V., Rousset, M.: The use of CARIN language and algorithms for information integration: the PICSEL system. Int. Jour. of Cooperative Information Systems (IJCIS) **9** (2000) 383–401
6. Harinarayan, V., Rajaraman, A., Ullman, J.: Implementing data cubes effciently. In: IEEE Transactions on Data Engineering. (1996) 1–25

7. Codd, E.: Providing olap (on-line analytical processing) to user-analysts: an it mandate. Technical report, E.F. Codd and Associates (1993)
8. Chaudhuri, S., Dayal, U.: An overview of data warehousing and olap technology. In ACM-SIGMOD **Record 26(1)** (1997)
9. Hassas, S., Fenet, S.: Ant: a distributed problem solving framework based on mobile agents. Advances in Mobile Ag. Sys. Research **1** (2000)
10. : Leap's official website. http://leap.crm-paris.com (2002)
11. Burg, B.: Towards the deployment of an open agent world. In: Journées Francophones d'Intelligence Artificielle Distribuée et de Systèmes Multi-Agents (JFIADSMA). (2000) 1–17
12. Sykara, K., Zeng, D.: Cooperative intelligent software agents. Technical Report CMU-RI-TR-95-14, Robotics Institut T.R. (1995)
13. Sykara, K., Decker, K., Pannu, A., Williamson, M., Zeng, D.: Distributed intelligent agents. In: IEEE Expert. (1996) 1–32
14. FIPA: The Foundation for Intelligent Physical Agents. http://www.fipa.org (2002)
15. Kimball, R., Merz, R.: The data webhousing. Eyrolles (2000)
16. Goglin, J.: The Data Warehouse building: From Data mart to Data web. Collection nouvelles technologies informatique. Hermes (1998)
17. Darmont, J., Boussaid, O., Bentayeb, F., Rabaseda, S., Zellouf, Y. In: Web multiform data structuring for warehousing. Volume 22 of Multimedia Systems and Applications. Kluwer Academic Publishers (2002) 179–194 Book title: Multimedia Mining: A Highway to Intelligent Multimedia Documents; Editor: Chabane Djeraba.
18. Miniaoui, S., Darmont, J., Boussaid, O.: Web data modeling for integration in data warehouses. In: First International Workshop on Multimedia Data and Document Engineering (MDDE 01), Lyon (2001) 88–97
19. Darmont, J., Boussaid, O., Bentayeb, F.: Warehousing web data. In: 4th International Conference on Information Integration and Web-based Applications and Services (iiWAS 02), Bandung, Indonesia, SCS Europe Bvba (2002) 148–152
20. Klusch, M.: Information agent technology for the internet: A survey. Journal on Data and Knowledge Engineering, Special Issue on Intelligent Information Integration, D. Fensel (Ed.) **36** (2001)
21. Sun Microsystems: Java Technology. http://java.sun.com (2002)
22. JADE: The java agent development framework. http://sharon.cselt.it/projects/jade/ (2002)
23. Bellifemine, F., Poggi, A., Rimassa, G.: Jade: a fipa compliant agent framework. In: PAAM'99, London. (1999) 97–108
24. Pitt, J., Bellifemine, F.: A protocol-based semantics for fipa 97 acl and its implementation in jade. In: AI*IA'99. (1999)
25. Andjomshoa, A., Shafazand, M., S.Bahonar: The application of software agent technology to project management infrastructure. In: 4th International Conference on Information Integration and Web-based Applications and Services (iiWAS'02), Bandung, Indonesia (2002)

A Multi-agent Architecture for Distributed Design

Ovidiu Chira[1], Camelia Chira[1], David Tormey[2], Attracta Brennan[2], and Thomas Roche[1]

[1]Galway-Mayo Institute of Technology, Dublin Road, Galway, Ireland
ovichira@yahoo.com, tom.roche@gmit.ie,
camelia.chira@nuigalway.ie
[2]National University of Ireland, CIMRU, Nuns Island, Galway, Ireland
abrennan@geminga.it.nuigalway.ie, david.tormey@nuigalway.ie

Abstract. Due to current trends in the design field towards virtual teams that collaborate over computer networks to achieve global optima in design, there is an increasing need for design teams to establish and maintain a cooperative work through effective communication, co-location, coordination and collaboration at the knowledge level. As problems become more complex, teamwork is becoming increasingly important. This paper *proposes* a multi-agent architecture to support multidisciplinary design teams that cooperate in a distributed design environment. Using ontologies and multi-agent systems, the proposed framework addresses resource allocation problems and aims to optimise design process operation and management.

1 Introduction

New models of the enterprise (such as the extended enterprise) involve multiple organisations cooperating together to design, manufacture and market products. In the context of design, multiple users with concurrent access to multiple system resources providing quick access to high quality information need to be supported. Distributed design environments (DDE) represent an answer to market demands and also an approach to a solution to specific design problems (see section two). One immediate benefit of collaborative work is the coming together of participants with heterogeneous skills [1], who, on sharing their skills, expertise and insight, create what is known as distributed cognition. "Distributed cognition emphasizes that the heart of intelligent human performance is not the individual human mind in isolation but the interaction of the mind with tools and artifacts as well as groups of minds in interaction with each other" [2]. DDE aims also include savings in project life-cycle and costs, added value to team efforts, access to a comprehensive knowledge-based system, reliable communication among design teams and members, flexible access and retrieval of information and timely connectivity with global experts [3-5]. However, because of the distributed nature of users and information resources involved in the design process, cooperative multidisciplinary design teams dispersed across the enterprise have to be supported and, the management of distributed

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 213–224, 2003.

information and knowledge has to be facilitated. Current DDEs fall short of providing an effective solution to distributed cognition.

This paper proposes a multi-agent architecture to address the problems that can arise during the collaboration process between distributed individuals in a virtual environment. Characterised by autonomy, pro-activeness and reactivity, software agents and multi-agent systems represent a potential solution to these complex distributed design problems. Using ontologies for knowledge sharing, reuse and integration and multi-agent systems for enabling interoperation among distributed resources, the proposed architecture is intended to facilitate the access, retrieval, exchange and presentation of data, information, and knowledge (both visually and aurally) to distributed design teams, in such a way that their collective conceptual space is expanded, learning and creativity strategies are supported and design solutions are optimised. Indeed, the agent and multi-agent system approach can deal with complex problem solving processes and is appropriate for domains in which data, control, expertise and/or resources are inherently distributed [6-8].

In this paper some major distributed design problems are identified; ontologies, agents and multi-agent systems are reviewed and then applied in order to specify the proposed multi-agent architecture. Finally, conclusions and future work are drawn at the end of the paper.

2 Distributed Design Problems

Global competition is forcing manufacturers to develop increasingly complex products in ever decreasing times [9-12]. Often it is beyond the scope of a department to fully contribute to their design. While distributed teams and the availability and communication of knowledge remain key success factors, they also represent one of the biggest obstacles for design in the future [4, 13]. Any individual, or group of individuals, involved in a product development process within the extended enterprise, (e.g. designers, product manufacturers, suppliers and design information providers) are participants in a DDE. The distribution of human and information resources involved in a DDE can be *geographical*, *temporal*, *functional* (i.e. the users are structured in clusters defined by specific perceptual, effectual and intellectual capabilities) and/or *semantic* (i.e. the users are structured in clusters defined by specific languages and conceptual realities). From a design perspective, “complex design problems generally require more knowledge that any one single person possesses because the knowledge relevant to a problem is usually distributed among stakeholders” [2]. Furthermore, designers working on one specific area of a project generally have a limited awareness of how the work of other designers working in the same project impacts on their own work.

Because of the volume and dispersion of knowledge, it has become a challenge for designers in traditional organizations to work together in teams in order to achieve a consensus based optimality [4]. With dispersed design personnel increasingly coming together to work collaboratively on design problems, major issues can arise. These can include firstly, how to synthesis potentially disparate perspectives on a problem and secondly how to manage large amounts of information [14]. With regard to the

latter issue, designers are usually bombarded by the plentiful supply of readily available information. As a result, the increasingly critical problem for them is to find that information which is relevant to the task at hand [15]. From a designer' perspective, the support for information access is vital. Hence, designers need support mechanisms to synthesise appropriate information from across the enterprise.

Furthermore, evolving information and knowledge residing within an enterprise is typically captured in databases and case bases which have to be trawled regularly by the designer, (research has shown that 47% of a designer's time is spent retrieving and managing information [16]). While this data is made accessible via a graphical user interface, current user interfaces only respond to direct manipulation, i.e. the computer is passive and always waits to execute highly specified instructions from the user. It provides little or no proactive help for the complex task of carrying out actions such as searches for information that may take an indefinite length of time. Furthermore, descriptive models of the design process suggest that design engineers are goal focused, tend to pursue a single design concept and attempt to optimise their original idea rather than generate new ones [17, 18]. Additionally if information is not easily accessible design engineers are unlikely to seek or share knowledge and expertise and as a result, at best are likely to generate local rather than global optima [17]. These problems have the potential to be particularly augmented in the extended enterprise environment because of the distribution of information and knowledge, the inherent dynamic nature of design information, the virtual communication processes and the increased complexity of products [19]. Extensive work has been carried out on passive information management systems for design, however, very little work has been carried out in the much needed area of proactive, holistic information management and presentation systems required by individuals within the extended enterprise.

To summarize, the authors believe that while the data/information/knowledge hierarchy is the main asset within a DDE, it also represents the most elusive resource to manage. Playing a critical role within this hierarchy, knowledge must be organised and managed so that human access to it is supported. Michael Polanyi and Ikijuro Nonaka indicate that knowledge has two forms i.e. implicit knowledge and explicit knowledge [20-22]. *Implicit* or *tacit* knowledge represents personal knowledge stored in the bearer's mental structures, having its roots in the private psychological baggage of the individual (e.g. subjective insights, intuitions and hunches). This kind of knowledge cannot be easily formalised, hence it cannot be straightforwardly communicated (at least not by the means of a formal language) or shared. *Explicit knowledge* is the knowledge codified and systematically expressed in formal structures compatible with human language (e.g. libraries, archives, databases). Hence, the explicit knowledge represents the kind of knowledge that is communicated and shared. Indeed, these two types of knowledge are crucial to the design process as a whole. Internalised knowledge is more likely to influence the designer subliminally in making good design decisions [23] and is also involved in the learning process. Research has shown that there are strong impacts of learning on the decisions made by the designer [23]. It is widely believed that "we can know more than we can tell" [22], and therefore a complete knowledge management system should aim to manage effectively the explicit knowledge as well as to support, promote and enable human's implicit knowledge, as this will affect creativity.

3 Multi-agent Systems and Ontologies

Started in the early eighties, software agents and multi-agent systems (MAS) research is one of the most active and fast growing areas of Artificial Intelligence (AI) and more generally of Computer Science [24-28]. Jennings suggests the necessity of using autonomous agents for developing robust and scalable software systems based on two arguments i.e. the adequacy hypothesis and the establishment hypothesis [27]. Also, Wooldridge and Ciancarini [28] indicate that intelligent agents and MAS systems represent techniques to manage the complexity inherent in software systems and can be considered an important new direction in software engineering [27, 28].

Although many and varied definitions for the notion of agency have been given [6, 24, 29-34], most researchers agree that a software agent is a computer system situated in an environment (and able to perceive that environment) that autonomously acts on behalf of its user, has a set of objectives and takes actions in order to accomplish these objectives [24, 27, 32]. *Autonomy* is the most important property of an agent without which the notion of agency would not exist. Autonomous agents can take decisions without the intervention of humans or other systems based on the individual state and goals of the agent. Furthermore, many researchers consider that an agent should also be characterised by one or more of the following properties [24, 25, 30, 32]: *reactivity* (an agent is situated in an environment and is able to perceive this environment and to respond to changes that occur in it), *pro-activeness* (an agent should have the ability to take the initiative in order to pursue its individual goals), *cooperation* (an agent should have the capability of interacting with other agents and possibly humans via an agent-communication language), *learning* (an agent should have the ability to learn while acting and reacting in its environment), *mobility* (a mobile agent has the ability to move around a network in a self-directed way) and *temporal continuity* (the actions of an agent are performed through a continuous running process).

MAS researchers study the behavior of a group of autonomous agents which are working together towards a common goal. Ideal for solving complex problems with multiple solving methods, perspectives and/or problem solving entities, MAS systems present many potential advantages including robustness, efficiency, flexibility, adaptivity, scalability, inter-operation of multiple existing legacy systems, enhanced speed, reliability and extensibility [6, 25, 26, 35, 36]. A MAS system can be defined as a "loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver" [6]. The problem solvers from this definition are autonomous and possibly heterogeneous agents. Since agents in a MAS system have to exchange information and knowledge in order to solve a problem coherently, *coordination*, *negotiation* and *communication* areas have become of crucial importance in MAS research [26]. The agents in a MAS system must *coordinate* their activities (to determine the organisational structure in a group of agents and to allocate tasks and resources), *negotiate* if a conflict occurs and be able to *communicate* with other agents. An agent communication language (ACL) enables agents to collaborate with each other providing them with the means of exchanging information and knowledge [37]. Besides an ACL, a common understanding of the concepts used among agents is necessary for a meaningful agent communication. This is because agents may have different terms for the same concept or identical terms for different concepts [38].

Therefore, ontologies are used for representing the knowledge from various application domains. The ACL remains just syntax without a shared common ontology containing the terms used in agent communication and the knowledge (e.g. definitions, attributes, relationships between terms and constraints) associated with them [39].

The study of ontologies has developed gradually from specific needs associated with the problem of knowledge management within a computational environment and particularly from the problem of knowledge sharing and reuse. Ontologies overcome the difficulties raised by "monolithic, isolated knowledge systems" [40], by specifying content specific agreements to facilitate knowledge sharing and reuse among systems that submit to the same ontology/ontologies by the means of ontological commitments [41, 42]. They describe concepts and relations assumed to be always true independent from a particular domain by a community of humans and/or agents that commit to that view of the world [43]. A merge of Gruber [44] and Borst et al [45] definitions is generally accepted by researchers, as follows: "Ontologies are explicit formal specification of a shared conceptualization" [46], where *explicit* means that "the type of concepts used, and the constraints on their use are explicitly defined", *formal* means that "the ontology should be machine readable, which excludes natural language", *shared* "reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group" and *conceptualization* emphasizes the "abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon". A detailed review of the different understandings for the concept of ontology and explanations can be found in [47, 48].

Ontologies are currently very popular mainly within fields that require a knowledge-intensive approach to their methodologies and system development, such as knowledge engineering [44, 49-51], knowledge representation [50, 52], qualitative modeling, language engineering, database design [53], information modeling [54], information integration [50, 55, 56], knowledge management and organization and agent-based design [38, 39, 57].

4 The Proposed Multi-agent Architecture

A multi-agent architecture called IDIMS (**I**ntelligent Agent Based Collaborative **D**esign **I**nformation **M**anagement and **S**upport Architecture) is proposed to support the designer's decision-making process, to facilitate the interoperation among DDE participants during the various design activities and to manage design knowledge. The goal of the IDIMS architecture is twofold as follows: (i) *to minimise* the effect of users and resources dispersion (particularly the temporal and geographical) and of the misunderstandings that might be generated by the (otherwise beneficial) functional and semantic distribution, the time spent for searching and retrieval of information, the effort of information translation between different tools and the administrational and organisational efforts not directly related to the design process (e.g. revision control) and (ii) *to maximise* the quality of information, knowledge sharing and reuse, the flexibility of the user interfaces, designer's learning curve and creativity, and the openness to the adoption of new strategies (e.g. Design For Environment strategies).

4.1 The Multi-agent Structure

Subsequent to the distributed design problems identified in section two, the MAS solution has been adopted to achieve the proposed goals of the IDIMS architecture (see Fig. 1).

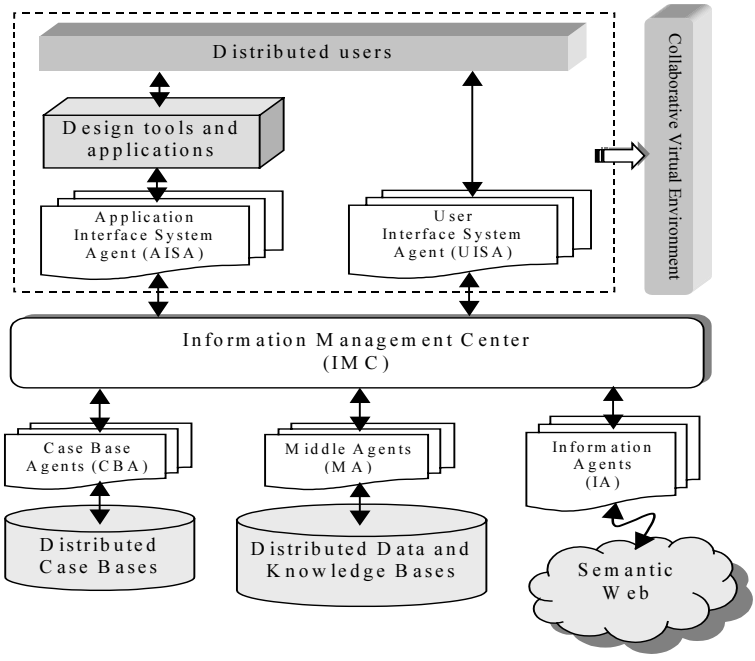


Fig. 1. An agent-based architecture for distributed design

Enabled by a MAS system, the IDIMS architecture supports distributed users collocated within the Collaborative Virtual Environment and semantically integrated through an agent-based system called the Information Management Center. The structural strength of the proposed architecture relies upon six agent subsystems as follows: (1) Application Interface System Agent, (2) User Interface System Agent, (3) Case Base Agents, (4) Middle Agents, (5) Information Agents and (6) Information Management Center.

The *Application Interface System Agent (AISA)* is a set of agents consisting of a supervisory agent and a collection of highly autonomous agents that capture application specific data (e.g. from CAD tools) and translate it to an internal format as defined by the ontology library. The process can be reversed for third party tools that allow this. The supervisory agent holds specific information about certain design tools and applications and its main function is to initiate and coordinate AISA agents. The AISA agents have to communicate with the supervisory agent in order to obtain application specific information and therefore meet their design objectives.

The *User Interface System Agent (UISA)* is similar to AISA but the UISA agents deal with any user specific aspect within a DDE e.g. collaboration with other DDE participants by providing an interface to the Collaborative Virtual Environment, captures of/requests for case bases or knowledge. They are tailored (and able to model themselves through learning) according to specific user needs and preferences and act autonomously in DDE. Both AISA and UISA supervisory agents control the base behaviour of their agents so that any required change of behaviour is made once and inherited by all agents.

The *Case Base Agents (CBA)* are responsible for distributed case bases management. They have the ability to store and retrieve case bases as well as to perform consistency checking and revision control.

The *Middle Agents (MA)* manage the distributed data/information/knowledge hierarchy e.g. storing, structuring (according to the ontology library), correcting, updating, maintaining, retrieving and consistency checking of data, information and/or knowledge. This management activity is a bi-directional one, i.e. MA can extend the knowledge base from input data or information and can also identify and extract appropriate data and information from knowledge.

The *Information Agents (IA)* exploit the vast amount of information available in wide area networks e.g. Internet and retrieve specific required information. These agents will reach their true potential when the web will be semantically enabled. The current trend of research suggests that the World Wide Web will be upgraded to a new environment i.e. Semantic Web where data will be defined and linked in such a way that it can be used by people and processed by machines [58-64].

Staying at the core of the MAS architecture, the *Information Management Center (IMC)* handles requests generated by different AISA and UISA agents and, in turn, generates requests for CBA, MA and IA agents. It consists of a set of mobile agents (i.e. agents that have the ability to move around a network in a self-directed way [24, 25]), supervised by one or more *coordinator agents*, that handle the request-response process. These mobile agents travel through the DDE network to regularly check for requests and forward them to a FIFO (First In First Out) structure. The coordinator agent(s) will organise the requests on three categories aimed for the CBA, MA and IA respectively. Once a response has been generated, the mobile agents will forward it from CBA, MA or IA to the requester.

In summary, the MAS architecture within the IDIMS system consists of five autonomous subsystems i.e. AISA, UISA, CBA, MA and IA, each of which dealing with clearly defined aspects of DDE. A sixth agent subsystem i.e. IMC integrates them by handling the intersystem agent communication and cooperation. All agents are characterised by goal-directed behaviour but they have a local view of the environment except for the IMC agents, which have a more holistic view over the entire DDE. This is achieved through knowledge of all the communication protocols used in the system, the capability of translation between these protocols (a global inter-agent communication can be obtained in this way) and knowledge of the system topology.

4.2 Ontological Foundation

The interoperation of multiple agents within the IDIMS system is achieved through an ACL and a common shared ontology library. Fig. 2 shows the ontological part of the IDIMS architecture. The ontology library creates a common (i.e. design semantics are the same for all agents that commit to the ontology library), formal (i.e. design semantics are agent enabled) understanding of the design domain. This machine-readable pool of data, information and knowledge represents the environment in which multiple agents act within the IDIMS system.

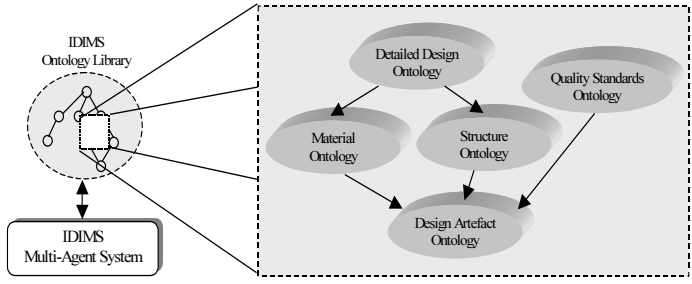


Fig. 2. Example of ontologies within the IDIMS system

A *Generic Design Ontology* (currently evolving) resides at the top-level of the ontology library. It introduces and defines the main concepts of the distributed design domain e.g. space, time, activity, process and artefact. The other ontologies are specializations of the Generic Design Ontology. Fig. 2 demonstrates some of the ontologies used by the IDIMS system. The *Detailed Design Ontology* formalizes general concepts specific to the detailed design phase. The *Quality Standards Ontology* defines the various quality standards and techniques (e.g. ISO9000, FMEA – Failure Mode Effect Analysis, TQM – Total Quality Management) that might be used throughout the design process, so that the artefact will adhere to certain quality standards. The *Material Ontology* defines concepts and relations about the material properties relating to an artefact (e.g. material type, material ID, ductility, malleability, thermal conductivity and density). The *Structure Ontology* describes the relationships between the components of the artefact (e.g. fasteners, assembly/disassembly times, routes and tools). Both the Material and Structure Ontologies are specializations of the Detailed Design Ontology. The *Design Artefact Ontology* is a further specialization of ontologies such as the Material Ontology and the Structure Ontology and will describe the various design parameters of the artefact.

To exemplify some of the information contained in the Design Artefact Ontology as a *specialization* of the Material and Structure Ontologies, Fig. 3 presents a simple case study of a smoke alarm product. Represented in the figure by shadowed boxes, the Product, Subassembly, Part and Fastener classes are defined within the Structure Ontology while the Material class is defined in the Material Ontology. Being a specialization of the concepts, relationships and rules defined in the Structure

Ontology, the smoke alarm product is represented using instantiated properties such as isPartOf, usesFastener and hasMaterial.

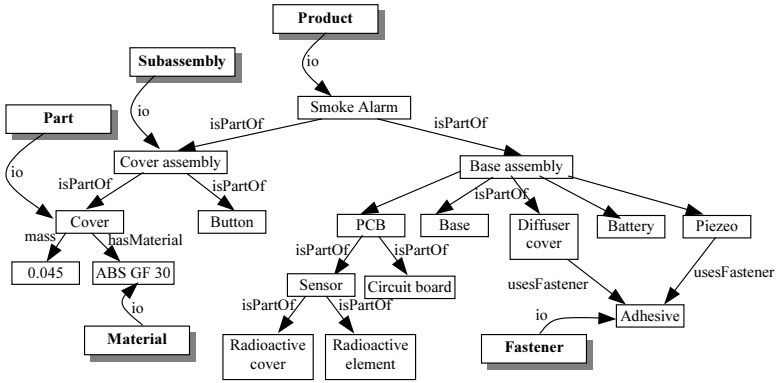


Fig. 3. Design artefact ontology for the smoke alarm (io stands for "instance of"). For simplicity reasons, only one fastener (i.e. Adhesive between the diffuser cover and the piezeo) within the smoke alarm and two part properties (i.e. mass and material) for one of the parts (i.e. cover) are represented.

Ongoing research and development work focuses on formalising the distributed design process by developing design specific ontology modules for the Process Specification Language (PSL –see http://ats.nist.gov/psl/). Having the functionality of the different IDIMS components identified, the proposed multi-agent architecture is currently designed and implemented using the Unified Modeling Language (UML) or possible UML agent extensions and the Java programming language.

5 Conclusions and Future Work

Being highly knowledge-oriented, distributed design environments require complex decision making processes. Therefore, the need for appropriate knowledge management tools is imperative. The proposed multi-agent architecture is intended to facilitate the management of the data-information-knowledge value chain efficiently in order to meet design organizational and operational goals (i.e. the global optimum).

The distributed design problems identified earlier in this paper are summarised under three interrelated headings, i.e. knowledge, user and infrastructure. The knowledge aspect is addressed by the AISA, CBA, MA and IA modules within the IDIMS multi-agent architecture. The user aspect (e.g. human-computer interaction, collocation within DDE, creativity enhancing, learning) is primarily managed by the UISA while the infrastructure aspect (e.g. heterogeneous platforms, applications and tools, user distribution) is tackled by the AISA, UISA and IMC modules.

Future work focuses on three main directions: (i) the testing, validation and improvement of the proposed multi-agent system for identifying the most suitable agent architecture and multi-agent communication, coordination and negotiation strategies, (ii) ontological alignment to current design standards (standards will more

than likely be proposed), (iii) research into human-computer interaction, collaborative virtual environments, information visualisation and particularly the user interface design.

While standard technologies usually offer pinpoint solutions (because they are dealing with data and rarely with information), ontologies and MAS systems represent a novel approach to address the problems created by highly knowledge-oriented environments such as distributed design. Capable of managing and processing not only data and information but also knowledge, the multi-agent solution takes a more holistic approach to the design environment supporting the designer's decision making process towards a global optima.

References

1. Edmonds E.A., Candy L., Jones R.,Soufi B., Support for Collaborative Design : Agents and Emergence. Communications of the ACM. 37(7):(1994).
2. Arias E., Eden H.,Fischer G., A. Gorman and E. Scharff. Transcending the Individual Human Mind - Creating Shared Understanding through Collaborative Design. ACM transactions on Computer-Human Interaction. Vol. 7 (2000) 84–113.
3. Iheagwara C.,Blyth A., Evaluation of the performance of ID systems in a switched and distributed environment the RealSecure case study. Computer Networks:(2002).
4. Pena-Mora F., Hussein K., Vadhavkar S.,Benjamin K., CAIRO: a Concurrent Engineering Meeting Environment for Virtual Design Teams. Artificial Intelligence in Engineering: (2000) 202–219.
5. Laure E., OpusJava: A Java framework for distributed high performance computing. Future Generation Computer Systems. 18:(2001) 235–251.
6. Jennings N.R., Sycara K.P.,Wooldridge M., A Roadmap of Agent Reasearch and Development. Journal of Autonomous Agents and Multi-Agent Systems. (1998) 7–36.
7. Oliveira E., Fischer K.,Stepankova O., Multi-agent systems: which research for which applications. Robotics and Autonomous Systems. 27:(1999) 91–106.
8. Wooldridge M., Agent-based computing. Interoperable Communication Networks. 1(1):(1998) 71–97.
9. Thoben K.D. Extended Products: Evolving Traditional Product Concepts. in 7th International Conference on Concurrent Enterprising. 2002.
10. Hirsch B., Extended Products in Dynamic Enterprises", E-Business: Key Issues, Applications and Technologies (2000) 622–628.
11. Kimura F. Inverse manufacturing: From Products to Services. in Managing Enterprises - Stakeholders, Engineering, Logistics and Achievement First International Conference Proceedings. (1997). MEP Ltd, London.
12. Tomiyama T., The Technical Concept of Intelligent Manufacturing Systems (IMS), University of Tokyo: Tokyo. (1994).
13. Agah A., Intelligent Graphical User Interfaces Design using Multiple Fuzzy Agents. Interacting With Computers. 12(5):(2000) 529–542.
14. Fischer G., Knowledge Management : Problems, Promises, Realities and Challenges. IEEE Intelligent Systems:(2002).
15. Viano G. Adaptive User Interface for Process Control based on Multi-Agent approach. in AVI 2000. (2000). Palermo, Italy.
16. Hales C., Analysis of the Engineering Design Process in an Industrial Context, in Department of Engineering University of Cambridge: Cambridge. (1987).
17. Lawson B., How Designers Think 2nd Ed, ed. B. Architecture (1990).

18. Hubka V., Design Science: Springer-Verlag (1996).
19. Roche T., Development of a Design for the Environment Workbench, in CIMRU, Industrial Engineering Dept UCG: Galway. (1999).
20. Nonaka I.,Konno N., The Concept of "Ba": Building a Foundation for Knowledge Creation. California Management Review. 40(3):(1998) 40–54.
21. Nonaka I.,Takeuchi H., The Knowledge Creating Company: How Japanese Companies Create the Dynasties of Innovation, Oxford University Press. (1995).
22. Polanyi M., The Tacit Dimension: Doubleday & Co. (1966).
23. Kolb D., Experiential Learning: Experience as the Source of Learning and Development: Prentice-Hall. (1984).
24. Nwana H.S., Software Agents: An Overview. Knowledge Engineering Review. 11(3):(1996) 1–40.
25. Bradshow J.M., An Introduction to Software Agents, in Software Agents, J.M. Bradshow, Editor. (1997), MIT Press: Cambridge.
26. Green S., Hurst L., Nangle B., Cunningham P., Somers F.,Evans R., Software Agents: A review, Intelligent Agents Group, Trinity College Dublin, Broadcom Eireann Research Ltd.: Dublin. (1997).
27. Jennings N.R., On agend-based software engineering. Artficial Intelligence:(2000).
28. Wooldridge M.,Ciancarini P., Agent-Oriented Software Engineering: The State of the Art, In P. Ciancarini and M. Wooldridge (eds.), Agent-Oriented Software Engineering, Springer-Verlag (2001).
29. Maes P., Artificial Life meets Entertainment: Lifelike Autonomous Agents. Communications of the ACM, ACM Press. 38(11):(1995) 108–114.
30. Franklin S.,Graesser A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. in Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, (1996).
31. Shoham Y., Agent-oriented programming, in Readings in Agents. (1998).
32. Wooldridge M., Intelligent Agents. An Introduction to Multiagent Systems, ed. G. Weiss: The MIT Press. (1999).
33. Poslad S., Buckle P.,Hadingham R. The FIPA-OS Agent Platform: Open Source for Open Standards. in Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents. (2000).
34. Russell S.,Norvig P., Artificial Intelligence: A Modern Approach, 2/E: Prentice Hall.(2003).
35. Gasser L., Social conceptions of knowledge and action: DAI foundations and open systems dynamics, In M.N. Huhns and M.P. Singh (eds.), Readings in Agents (1998).
36. Martin F.J., Plaza E., Rodriguez-Aguilar J.A.,Sabater J. Java Interagents for Multi-Agent Systems. in Software Tools for Developing Agents. (1998).
37. Labrou Y., Finin T.,Peng Y., Agent Communication Languages: The Current Landscape. IEEE Intelligent Systems: (1999).
38. Odell J., Agent Technology, OMG - Agent Platform Special Interest Group. (2000).
39. Nwana H.,Wooldridge M., Software Agent Technologies. BT Technology Journal. 14(4):(1996) 68–78.
40. Gruber T.R. The Role of Common Ontology in Achieving Shareable, Reusable Knowledge Bases. in Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (1991).
41. Spyns P., Meersman R.,Jarrar M., Data Modelling versus Ontology Engineering, ACM SIGMOD Record. (2002).
42. Gruber T.R., Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human and Computer Studies.(1995) 907–928.

43. Guarino N. Semantic Matching:Formal Ontological Distinctions for Information Organization, Extraction, and Integration. in Summer School on Information Extraction. (1997). Frascati, Italy, July 14–19.
44. Gruber T.R., A Translation Approach to Portable Ontology Specification. Knowledge Aquisition. 5(2):(1993) 199–220.
45. Borst P., Akkermans H.,Top J., Engineering Ontologies. International Journal of Human-Computer Studies. 46(Special Issue on Using Explicit Ontologies in KBS Development): (1997) 365–406.
46. Studer R., Benjamins V.R.,Fensel D., Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering. 25(1-2):(1998) 161–197.
47. Guarino N., Understanding, Building and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development. International Journal of Human and Computer Studies. 46:(1997) 293–310.
48. Guarino N.,Giaretta P., Ontologies and Knowledge Bases: Towards a Terminological Clarification, in Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, N. Mars, Editor. (1995), IOS Press: Amsterdam. 25–32.
49. Gaines B., Editorial: Using Explicite Ontologies in Knowledge-based System Development. International Journal of Human-Computer Systems. 46:(1997).
50. Guarino N. Formal Ontology and Information Systems. in Formal Ontology in Information Systems. FOIS'98, 6–8 June 1998. (1998). Trento: IOS Press.
51. Uschold M.,Gruninger M., Ontologies:Principles, Methods and Applications. The Knowledge Engineering Review. 11(2):(1996) 93–136.
52. Artala A., Franconi E., Guarino N.,Pazzi L., Part-Whole Relations in Object-Centered Systems: an Overview. Data and Knowledge Engineering. 20(3):(1996) 347–383.
53. Van de Riet R., Burg, H., Dehne, F., Linguistic Issues in Information System Design, In G. Nicola (eds.), Formal Ontology in Information System (1998).
54. Weber R., Ontological Foundations of Information Systems. Melbourne: Coopers and Lybrand.(1997).
55. Mena E., Kashyap, V., Illarramendi, A., Sheth, A., Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure, in Formal Ontology in Information Systems, N. Guarino, Editor. (1998) IOS Press: Amsterdam.
56. Bergamaschi S., Castano S., Vimercati S.D.C.d.,Vincini M., An Intelligent Approach to Information Integration, in Formal Ontology in Information System, N. Guarino, Editor. (1998), IOS Press: Amsterdam.
57. Chaib-draa B.,Dignum F., Trends in Agent Communication Language. Computational Intelligence. 18(2):(2002).
58. Berners-Lee T., Semantic Web Road Map, World Wide Web Consortium: http://www.w3.org/DesignIssues/Semantic.html. (1998).
59. Berners-Lee T., Hendler J.,Lassila O., The semantic web. Scientific American. 284(5):(2001) 34–43.
60. Dumbill E., Building the Semantic Web, XML.com, (2001).
61. Fensel D., Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Berlin: Springer. (2000).
62. Ramsdell J.D., A Foundation for a Semantic Web. (2000).
63. Decker S., Harmelen F.v., Broekstra J., Erdmann M., Fensel D., Horrocks I., Klein M.,Melnik S., The Semantic Web – on the respective Roles of XML and RDF. IEEE Internet Computing:(2000).
64. Hendler J., Berners-Lee T.,Miller E., Integrating Applications on the Semantic Web. Journal of the Institute of Electrical Engineers of Japan. 122(10):(2002) 676–680.

AgentAllocator: An Agent-Based Multi-criteria Decision Support System for Task Allocation

Nikolaos F. Matsatsinis and Pavlos Delias

Technical University of Crete
Decision Support Systems Laboratory
University Campus, 73100, Chania, Greece
{nikos, mirouvor}@ergasya.tuc.gr

Abstract. The multi-agent systems in Artificial Intelligence very often include task allocation problems. These problems are arduous to be modelled; therefore it is also difficult to end up with an optimal allocation plan. AgentAllocator is an easy to use, platform independent application, which implements a multi-criteria method to support the decision of Task Allocation. The decision maker is able to model the problem (according to his policy) through its inputs dialogs and employ the final solution proposed by the system. In this paper, both the theoretical background of AgentAllocator and the DSS itself are presented.

1 Introduction

The task allocation problem occurs very often in multiplex environments. This problem is considered as one of the most classical problems in the operational research, where the allocation is decided by validating a single optimization criterion (mono-criterion problem) or multiple optimization criteria (multi-criteria problem) [4],[5]. This paper attends to face the problem of task allocation in a multi-agent system. The allocation is based on the agent's attributes and on its ability to execute a specific task. The final decision comes upon the evaluation criteria used by the decision maker. A really significant feature of this model is the reevaluation of agents' attributes each time a task is assigned.

The method used by AgentAllocator is delineated in section 3, while in section 2 we refer to four different approaches for task allocation in multi-agent systems and we try to underline their basic differences from the proposed model. In Section 4, illustrative examples as well as a few screenshots of AgentAllocator are attached. Finally, in the last section we try to criticise both the method and the DSS, revealing our feature goals.

2 Background

The task allocation in multi-agent systems has occupied to a large extend the scientists of Decision Science and Artificial Intelligence.

Smith [12] proposed a "Contract Net Protocol" that agents can adopt. According to this approach, the agents negotiate during run time and come to a decision for the

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 225–235, 2003.

task allocation. Since agents are autonomous, the negotiation breadth can be really huge and the time needed until the agents come to decision may not be available. In addition, negotiations themselves demand continuous communication among the agents, and that means great cost.

In order to reduce the negotiation's breadth that is demanded from the "Contract Net Protocol", Tidhar, Rao and Sonneberg [13] proposed a more guided process of selecting a team of agents with complementary skills to achieve a goal. The instructor is the developer of the multi-agent system, because he/she knows a priori all the alternative agents and their attributes. Each time that a goal is to be achieved the instructor split it to sub-goals and for each sub-goal he/she specifies the attributes that an agent or a group of agents must have in order to bring it successfully to an end. In other words the instructor determines agents' types that could be candidate for a task allocation. If an agent does not belong to the definite for a task "type", he cannot participate in the negotiation that will take place for the assignment of this specific task. Consequently, the number of the candidates for a task allocation is significantly reduced and the allocation process is less time-consuming. Of course, since an agent or a group of agents is considered as a candidate for a specific task allocation and takes part in the negotiation procedure, he acts autonomously.

The total autonomy that Smith [12] as well as Tidhar et al. [13] assigned to agents causes in several occasions serious problems. Such an occasion is presented and faced by Walsh and Wellman [14]. In that case, the various agents can perform as suppliers of primitive goods or as producers of output goods and the goal is a consumer-agent to acquire the goods he desires. Each agent regularly sends bid messages for some of the goods that he wishes to buy or sell, specifying also the price below/above which the agent is willing to buy/sell. Each time that a new bid arises, the rest of the agents may choose to revise their own bids. Agents can respond only to the most recent bid sent to the system. Bidding continuous until a state where all messages have been received and no agent chooses to revise his bids. The problem that arises and that obviously the "Contract Net Protocol" approach can not face is that the bid messages, that agents send, have to be valid. For instance, a producer should not be able to claim that he will sell an output product if he has not ensured the purchase of the necessary primitive goods. Additionally, the bids have to follow a specific economy policy. To ensure these, it is necessary for the system to include several bidding policies, so that every agent adopts his favorite. This means that agents partially loose their autonomy and follow strictly several main rules.

In the above approaches, task allocation is brought into effect after a process of negotiation among the candidate agents themselves. A different approach is the one that indicates the existence of a "master-agent" who will be responsible for the allocation of the tasks to several other agents (slave agents). Such an approach is presented by Fujita and Lesser [2]. In that case there is a goal to achieve until a deadline and with a standard quality level. Firstly, the master-agent splits this goal to two or more sub-goals that must be achieved. There is the possibility that several sub-goals might need to be executed sequentially, meaning that the one's results are essential for the execution of the rest. Using heuristics algorithms the authors determine the order according to which the sub-goals will be allocated and executed. The sub-goals assignment to the slave-agents is based to a single criterion, the minimization of the central goal's execution duration. Specifically a slave-agent will

undertake the execution of a sub-goal if and only if he is the most immediate available among the other agents. In case that the execution of a sub-goal prerequisite for more than one other sub-goals, both of them will be assigned to the same slave-agent. Of course if a sub-goal is prerequisite for more than one of the other sub-goals, then the agent who has undertake it, will also undertake and one of the other sub-goals, while the rest will be allocated to agents who will be available by the time that their execution will be feasible (that is by the time that the prerequisite sub-goal will be completed). Finally, Fujita and Lesser [2] provide to the master agent the possibility to evaluate periodically the initial plan of tasks allocation, to ensure that the time and quality limits of the central goal are satisfied.

Some important works study the principles of an agent decision making function in a range of disciplines including economics, sociology and time restrictions. Traditionally this function has been based in the optimization of a single criterion. Thus the Agents Decision Makers (ADM) choice was based on the minimization of the performance time or the results accuracy. Another approach (Parsons and Jennings [8]) makes a more social confrontation. The candidate agents are ranked not only in the base of their individual skills, but also according to their social behavior (past cooperations). Moraitis and Tsoukias [7] propose a multicriteria approach, which considers a set of criteria of each candidate agent, and through direct pairwise comparisons establish a graph representation of the candidate agents' behavior. The above multicriteria approaches are based on value focused procedures and aim to construct a unique function, aggregating the partial preferences on multiple criteria. In that case the ADM is directed by this value system to its final decision. Shehory et al. [10] mention the problem of task allocation to a group of agents in order to attain tasks execution and to improve tasks efficiency. We will confront the problem from a different point of view.

In this paper a new method for task allocation is presented. There are no more master and slave agents or negotiation among candidate agents. The multi-agent system itself is responsible for the task allocation. The candidate agents are predefined. Each of them has attributes – capabilities while each task is described by specific attributes – demands. The system counts all the possible assignments and evaluates them all through a multi-criteria method described below. As a result the final allocation plan is as much as possible consistent with the decision maker's preferences.

3 The Multi-criteria Method

AgentAllocator uses the method proposed by Matsatsinis et al. [6]. This is about a value-focused approach established in three main functions:

1. Modeling a consistent family of measurable and/or ordinal criteria to evaluate the performances of every possible task assignment to agents.
2. Assessing an additive value function aggregating the evaluation criteria in order to provide a ranking of alternative assignments.
3. Implementing an allocation mechanism which finally assigns the task to agents according to their value collected and which reviews agents' attributes.

3.1 Defining the Method and Making the Concessions

The focus of the method is to assign k tasks to m agents while the loads of tasks may be smaller, equal or even greater of the crowd of agents. Obviously, every agent is eligible to undertake more than one task. The method declares that each task can be executed just by one agent and that since the allocation is decided the agent has no right to negotiate this decision. Moreover, agents do not advance any of the tasks.

At this point it is necessary to employ the following notations and definitions:

Let:

T	be the set of tasks
A	set of agents
k	number of the tasks
m	number of the agents
t_i	the i-th task in T
a_j	the j-th agent in A
(t_i,a_j)	assignment action of t_i to a_j
$v(T)$	set of demands describing T
$v(A)$	set of attributes describing A
F	a consistent family of criteria
N	number of these criteria
g_j	the j-th criterion in F
$g_j(t_i,a_l)$	the performance of (t_i,a_l) assignment on the j-th criterion
$G(t_i,a_l)$	vector of performances of (t_i,a_l) assignment on the N criteria
g_{j*}	worst level of criterion g_j
g_j^*	best level of criterion g_j
$u_j(g_j)$	a non decreasing marginal value function on $[g_{j*}, g_j^*]$ varying from 0 to 1
p_j	weighting factor of criterion g_j (the sum of N weights equals 1)
$U(G)$	the global value of the performance vector G, varying from 0 to 1.

3.2 The Task Allocation Process

The various tasks are not identical but all can be described through a common set of demands. The same stands for the agents. They are also distinct and can be described by a common set of attributes. The agents' attributes may change dynamically during the allocation process. They may be improving, reducing or even remain constant. Both these demands and attributes varying between several rating levels. These levels are preferably descriptive. All these sets are formulated by the decision maker and entered in the system as input files. The next critical step is to shape the evaluation criteria. The evaluation criteria have to be modeled in such a way that they compose a consistent family of criteria and (as knowing from the multi-criteria analysis theory) have to fulfill three fundamental conditions: monotony, exhaustively and non-pleonasm [10]. Each criterion is modeled by a set of sub-criteria, defining hereby a sub-criterion as a combination of task's demands and agent's attributes. Those sub-criteria have also rating levels varying from 0 to 1(a numerical correspondence is

attached to every descriptive level of the sub-criterion). Rating the sub-criteria, we have to ponder that:

- High performance agents should not be wasted in tasks with low demands.
- No sub-criterion should be advanced, unless it is a completely satisfactory one.
- All not workable assignments should be rated with a 0.

This is going to be more clear in the example that follows. Modeling the criteria may be the most critical step in this method as it incorporates the decision's maker logic and experience.

With a view to evaluate each assignment's performance the following heuristic process is executed:

1. Divide each criterion g_i to several level-characterizations and correspond each level with a numerical interval
2. For each assignment (t_i,a_j):
   - According to the content of each sub-criterion find the relevance between task's demands and agent's attributes and value (t_i,a_j) to the specific sub-criterion
   - For each criterion g_i sum up the values that (t_i,a_j) has achieved in the corresponding set of sub-criteria and calculate the average value
3. For each criterion g_i, give to each assignment the appropriate characterization based on the interval that the calculated average value belongs to.

Keeping on with the method, the decision maker is asked to rank a set of reference assignments, in such a way that the ranking reveals preference or indifference among the alternatives assignments. This ranking is used by the UTA [3], [11] method when assessing the criteria weights. UTA method is explained in section 3.3.

Applying this global utility function all assignments are being evaluated and given a score. The assignment that achieved the highest score is selected and the allocation is considered to be done. The task component of this assignment is removed from the set of task and the agents' attributes are reviewed according to the following single rule: Let q be the total of tasks that this specific agent has undertaken and let s be the total of the rating levels of each attribute. Then, for each attribute:

- If $q \geq P/s$, the attribute switch upwards or downwards its rating level
- If $q < P/s$, the attribute's rating level stands still, where P is a percentage defined by the decision maker.

The method acts exactly the same, in an iterating way, until the set of tasks is empty. Of course the assignments' scores are reevaluated in each step, since agents modify their attributes.

3.3 The UTA Method

UTA (**UT**ilité **A**dditive) method proposed by Siskos [11] tries to analyze the decision maker's preferences in a multicriteria environment while the alternatives decisions are predefined and finite. UTA provides a global additive utility function that represents the decision maker's preferences following the next steps:

Step 1: Give a demonstrative ranking of some reference alternatives

Step 2: Express the global values of the reference assignments in terms of weights:

$$U[g(A_i)] = \sum_{j=1}^{n} p_j u_j \left[g_j(A_i) \right] \qquad \forall\ i = 1, 2, \ldots, k \tag{1}$$

g_j stands for the criteria, p_j for criteria weights and A_i for the alternatives.

Step3: For each criterion g_j , j = 1,…,N (where N is the total of criteria), evaluate the ordinal marginal utility functions $u_j(g_j)$ in such a way that:
$u_i : \{g_{j*}, g_j^*\} \rightarrow [0,1]$, where g_{j*} is the worst and g_j^* the best value of the j-th criterion.
$u_i(g_{j*}) = 0$
$u_i(g_j^*) = 1$

Step 4: Introduce two errors σ_i^+ and σ_i^- per assignment A_i and formulate for each pair of consecutive assignments in the ranking, the analytical expressions:

$$\Delta\left(A, A_{+1}\right) = U[g(A)] - \sigma^+ + \sigma^- - \left[U[g(A_{+1})] - \sigma^+_{+1} + \sigma^-_{+1} \right] \tag{2}$$

The purpose and the significance of these errors is to represent the distance of each ranking preference given by the decision maker from the global utility function U.

Step5: Solve the linear program:

$$\text{Min } z = \sum_{i=1}^{k} (\sigma_i^+ + \sigma_i^-) \tag{3}$$

Subject to:
For i= 1,2,…k-1

$$\Delta(A_i, A_{i+1}) \geq \delta \text{ if } A_i > A_{i+1} \tag{4}$$
$$\Delta(A_i, A_{i+1}) = 0 \quad \text{if } A_i \approx A_{i+1}$$

$$\sum_{j=1}^{n} p_j = 1 \tag{5}$$

$$p_j \geq 0,\ j = 1,2,\ldots,n \tag{6}$$

$$\sigma_i^+ \geq 0,\ \sigma_i^- \geq 0,\ I = 1,2,\ldots,k \tag{7}$$

δ being a small positive number(AgentAllocator uses a default value of 0.05).

Step 6: (Stability Analysis) Test the existence of multiple optimal or near optimal solutions of the linear program (1)-(7); in case of non uniqueness, find the mean weights of those (near) optimal solutions which maximize the objective functions $z_j = p_j$ for all j = 1,2,…,n on the polyhedron (4)-(7) bounded by the new constraint:

$$\sum_{i=1}^{k}\left(\sigma_i^{+}+\sigma_i^{-}\right)\leq z^{*}+\varepsilon \tag{8}$$

where z^* being the optimal value of the linear program in step 3 and ε is a very small percentage of z^*. AgentAllocator uses a default value for ε 0.05 but the decision maker is eligible to change this percentage through the options menu of the DSS.

4 Applying AgentAllocator in an Illustrative Example

Let us monitor AgentAllocator during the following example. Consider an employer trying to manage his enterprise's executive officers-agents. For the purpose of this article, let three be the total number of these officers (we call them agents from now on). Agents have a certain profile, composed by the following attributes:

Table 1. Agents' Profiles.

Agent's name	Technical Knowledge	Problem Solving Ability	Reliability	Management Skills	Availability
a1	Basic	Undistinguished	Trustworthy	Clement	High
a2	Expert	Satisfactory	Unproved	Poor	Medium
a3	None	Satisfactory	Unproved	Expert	Medium

At this point, we should remind that agents have the ability to adjust dynamically their performances in certain attributes. So, in this case agents' *Availability* decreases and their *Management Skills* are worked up during run time. Let us also suppose that the tasks that occur in the enterprise environment can be described by the demands of: Immediacy, Importance, Technical Demands and their Social Minded aspect. All these demands have their own rating levels as shown in table 2.

Table 2. Tasks' requirements.

Task name	*Technical Demands*	*Immediacy*	*Importance*	*Social Minded*
t1	Basic	Normal	Normal	Normal
t2	Expert	Urgent	Normal	Normal
t3	Average	Urgent	High	Normal
t4	None	Urgent	High	Critical
t5	Basic	Low	High	Critical

The goal is to figure out the best allocation plan. For this reason, three evaluation criteria are employed: the *speediness* of the task's execution, the *risk* undertaken by each allocation decision and the *functionality* of the assignments. These criteria are modeled as declared in table 3:

Table 3. Modeling the Criteria.

Speediness	Risk	Functionality
Availability – Immediacy	Problem Solving Ability - Importance	Technical Knowledge-Technical Demands
Problem Solving Ability	Management Skills-Social Minded	Management Skills-Social Minded
	Reliability	

You may have noticed that the components of the sub-criteria are either agents' attributes or tasks' demands or both. For each sub-criterion, the decision maker has to correspond a numerical value at every possible combination of its components rating levels. Criteria, as mentioned in section 3, have their own rating levels. Table 4 gives an example of this sub-criteria rating procedure.

Table 4. Rating Sub-Criteria of Speediness Criterion.

Speediness Criterion					
			Availability		
	Immediacy		Low	Medium	High
		Low	1	0.6	0.4
		Medium	0.4	1	0.6
		Urgent	0	0.6	1
	Problem Solving Ability				
	Non Satisfactory		Undistinguished		Satisfactory
	0		0.4		1

All this data can be inserted in the system using its input dialogs. In figure 1, the input dialog for the agents' data is presented.

AgentAllocator includes four types of input dialogs. In Task Dialog, the user inserts the data needed to model the tasks, such us task's demands and its rating levels. In Agent Dialog, the decision maker should insert agent's attributes, attributes' cases (improving, reducing or constant) and their rating levels. Criteria Dialog is used to insert the data for the evaluation criteria, which means: criteria's intervals, sub-criteria and sub-criteria's rating levels. Finally, in Assignments Dialog, the user can watch the performance of all assignments in the evaluation criteria and insert the ranking for the set of a few reference assignments.

In order to define the global utility function, UTA method is implemented. UTA assesses the criteria weights analyzing a reference set of alternatives. In a reference set, ranking has the following meaning:

$$ranking(\text{Assignment}_i) > ranking(\text{Assignment}_j) \text{ indicates preference,}$$
$$ranking\ (\text{Assignment}_i) = ranking(\text{Assignment}_j) \text{ indicates indifference.}$$

The reference set used in this example is figured out in table 5.

Fig. 1 Agent's Data Input Dialog.

Table 5. Ranking of Reference Alternatives.

Preference Assignment	Rank	Risk	Speediness	Functionality
This is a Real Good Case!	1	Certainty	Big	Normal
Just Good	2	Admissible	Acceptable	Efficient
Not Satisfactory	3	Admissible	Big	Bad
Not Satisfactory too	3	Risky	Small	Normal
A Bad One	4	Risky	Acceptable	Bad

Given the above reference set, UTA assess *Functionality* a weight of 61.8% while *Speediness* achieved a 26.7% and *Risk* a 11.5%. AgentAllocator calculates the scores of all possible assignments and ends up with an optimal allocation plan. The scores of the remaining possible assignments are recalculated each time a task is allocated. These scores may differ from step to step because agents' attributes change

dynamically during the process. The user is able to watch the entire allocation process in the "task allocation board" which is attached to the system as represented below:

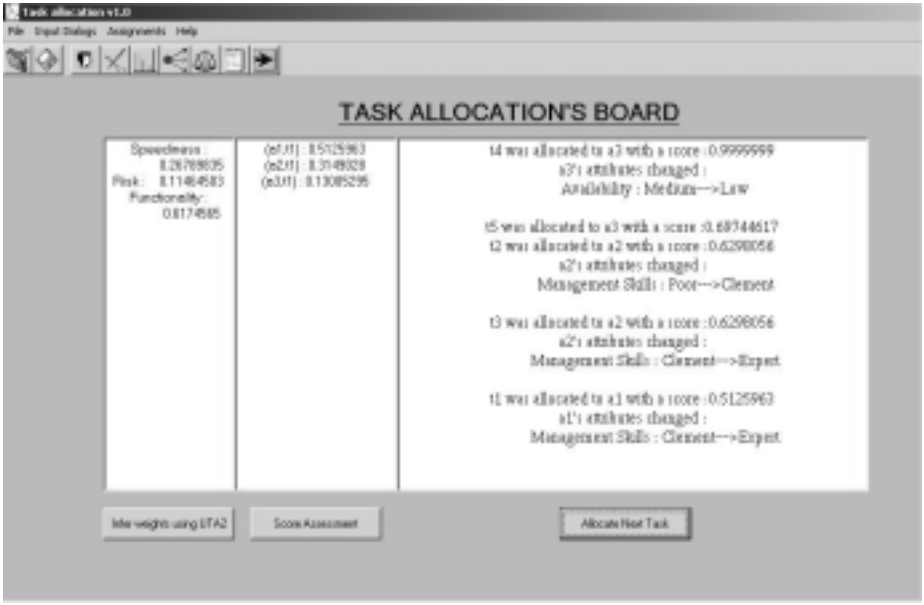


Fig. 2. Task Allocation's Board.

More specifically, the evolution of the task allocation process took place as follows:

Step1: The task called t4 is assigned to agent a3. The a3's attributes are reviewed and his *Availability* is reduced from *Medium* to *Low*. t4 is being removed from the set of tasks. The scores of the rest of the assignments are recalculated.

*Step 2:*t5 is allocated to a3.

Step3: t2 is allocated to a2. His *Management Skills* are shifted up to *Clement*.

*Step 4:*a2 undertakes t3. His *Management Skills* augments once more. a2's current *Management Skills* are rating with an *Expert* value. This double augmentation of a2's *Management Skills* can be avoided by increasing the Percentage *P* mentioned in section 3.2. But as long as this example is being used just for demonstration's reasons we shall allow to a2 this impressive evaluation in his *Management Skills!*

*Step 5:*One task has remained. That is t3 and it is assigned to a1. Now it is a1's turn to improve his attributes, so his *Management Skills* shift up to *Expert*.

5 Conclusions

The approach for task allocation proposed in this paper includes a multi-criteria methodology. That guides the final solution to be as possible consistent with the

decision maker's preferences. Therefore, the final solution depends in a very direct way on the decision maker's policy. This infuses the system with an impressive flexibility but also with a disagreeable subjectivity. To prevent this subjectivity from leading to dissatisfactory decisions, this approach should employ a way to measure the final allocation plan's effectiveness. There should be a feedback reporting the results of the selected allocation plan, so that the method could evaluate its own techniques. This feedback information would help even if it was implemented through every step of the allocation process, so the DSS could decide whether to improve more agents' attributes or to detract their improvement. A researching focus of this method is to model group allocation and some constraints that may task impose, such as precedence constraints. In addition, a future ambition is this DSS to include more allocation methods to support more efficiently the decision maker.

Concluding, AgentAllocator may subsist a compact and useful tool to model and analyze the task allocation decision in complex environments, which is hard to be modeled and analyzed otherwise.

References

1. Delias, P.: Design and Analysis of an Decision Support System Based on Agent Technology, Diploma Thesis, Technical University of Crete, (2002)
2. Fujita, S., Lesser, V.: Centralized Task Distribution in the Presence of Uncertainty and Time Deadlines, ICMAS-96, (1996)
3. Jacquet-Lagreze, E., Siskos, Y.: Assessing a set of additive utility functions for multi-criteria decision making, European Journal of Operational Research, 16,1, (1982), 48–56
4. Kenny, R.L., Raifa, H.: Decision with multiple objectives: Preferences and value tradeoffs, J.Wiley, NY, (1976)
5. Klein, G., Moskowitz, H., Maheesh, S., Ravindran, A.: Assessment of multi-attribute measurable value and utility functions via mathematical programming, Decision Sciences, 16,3, (1985), 309–324
6. Matsatsinis, N.F., Fostieri, M., Koutsouraki, E., Moraitis, P.: A Multi-Criteria Approach for Task Allocation in a Multi-Agent System, Proceedings of the ESIT'99, Chania, (1999)
7. Moraitis, P., Tsoukias, A.: Graph Representation of collective Attitudes for Teamwork in Dynamic Environment, (1998)
8. Parsons, S., Jennings, N.R.: Argumentation and multi-agent decision making, Proc. AAAI Spring Sym. on Interactive and Mixed-Initiative Decision Making, Stanford, USA, (1998), 89–91
9. Roy, B.: Methodologie multicritère d' Aide à la Decision, Economica, Paris, (1985)
10. Shehory, O., Kraus, S.: Task allocation via coalition formation among autonomous agents, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), (1995).
11. Siskos, Y.: Comment Modéliser les Preferences au Moyen de Functions d' Utilité Additives, RAIRO Recherche Opérationnelle, 14, (1980), 53–82
12. Smith, R.: The Contract Net Protocol: High Level Communication and Control in a Distributed Problem-solver, IEEE Transaction on Computers C-29(12), (1980), 1104–1113
13. Tidhar, G., Rao, S., Sonneberg, E.: Guided Team Selection, ICMAS-96, (1996)
14. Walsh, W., Wellman, M.: A Market Protocol for Decentralized Task Allocation, IEEE 1998, (1998)

An Approach to Process Automation Based on Cooperating Subprocess Agents

Ilkka Seilonen, Teppo Pirttioja, Pekka Appelqvist, Aarne Halme, and Kari Koskinen

Helsinki University of Technology, Computer and Information Systems in Automation, P.O.Box 5400, FIN-02015 HUT, Finland
{ilkka.seilonen, kari.o.koskinen}@hut.fi
http://www.automationit.hut.fi
Helsinki University of Technology, Automation Technology Laboratory, P.O.Box 5400, FIN-02015 HUT, Finland
{teppo.pirttioja, pekka.appelqvist, aarne.halme}@hut.fi
http://www.automation.hut.fi

Abstract. An approach to extend process automation systems with cooperating subprocess agents is presented in this paper. According to this approach a society of subprocess agents supervises an ordinary process automation system. The functionality of this agent layer includes supervising the lower-level automation system, semi-autonomous reconfiguration of its control logic when needed and query processing for external systems. In this way the approach aims for enhancing the operational flexibility of the automation system. The subprocess agents utilize several agent-based cooperation mechanisms in order to be able to perform their tasks. The approach is demonstrated with a laboratory test process where process startup and fault-recovery scenarios have been imitated. Experiences from initial test runs are described, too.

1 Introduction

This study is motivated by a potential match between the increasing requirements of process automation and the development of agent-based cooperation mechanisms. Although traditionally reliability, efficiency and quality have been the major requirements in process automation [15], recently also flexibility of operations and easiness of system maintenance have been emphasized [21]. Cooperation mechanisms developed in multi-agent systems (MAS) research [3], [23] have been assessed as potentially useful for this kind of requirements as indicated by research in other application domains, e.g. discrete manufacturing [12] and communication systems [7], [9]. Our long-term research objective is to specify a reasonable way to utilize MAS technology in order to improve process automation systems with the desired properties of enhanced operational flexibility and easiness of system maintenance.

The purpose of this paper is to present an approach to extend process automation systems with cooperating subprocess agents. Particularly, our aim is to specify the architecture and cooperation methods of a society of subprocess agents. The paper is

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 236–245, 2003.

outlined as follows. Chapter 2 will discuss cooperation requirements and MAS cooperation methods in the context of process automation. The architecture of the subprocess agent society is presented in Chapter 3. The cooperation mechanisms of the subprocess agents are described in Chapter 4. Our test environment and experiments with it are depicted in Chapter 5 followed by conclusions in Chapter 6.

2 Cooperation in Process Automation

Process automation systems have several functions with different characteristics including continuous, sequential and batch control, status and performance monitoring, and abnormal situation handling. Process automation systems are typically built as distributed systems that are integrated to external systems. There is both an internal and external need for cooperation between systems units in several functions. If MAS technology is going to be applied to the development of process of process automation systems, the distinct characteristics of cooperation in the different functions need to be taken into account.

Some researches have already studied the application of MAS technology to process automation systems [1], [16]. Agents have been used as modules of an automation system [22] and as an integration mechanism [2]. They have also been used for planning of batch control operations via Contract Net type of negotiation [11]. Recently also an approach using agent-based planning techniques for run-time specification of control sequences have been reported [13], [20]. This approach has been based on planning with centralized coordination and request-response type of communication. Also market-based negotiation has been studied in the context of closed loop control [24]. However, none of the mentioned studies has presented a comprehensive set of cooperation mechanisms needed for various functions of process automation.

Our approach introduces a few additions to the previous approaches. We emphasize the combination of several cooperation schemes with both deliberative and reactive behavior and utilization of several FIPA interaction protocols. Also in our cooperation models peer-to-peer coordination has more usage than in the mentioned approaches. Again, the cooperation mechanisms of MAS are applied not only to the control operations of subprocess agents but also to their information processing activities like query processing.

3 Architecture of the Subprocess Agent Society

According to our concept an extended process automation system consists of two layers as illustrated in Fig. 1 [17], [18]. A higher-level layer of subprocess agents supervises a lower-level, ordinary real-time process automation system. Both of these may be distributed systems. The agent layer is designed as a society of supervisory subprocess agents whose primary purpose is to monitor the lower-level automation system and reconfigure its operation logic cooperatively during run-time when

needed. An additional purpose of the agent layer is to cooperatively provide data from the process and automation system to external clients.

The agent layer consists of subprocess agents that communicate to each other according to the agent communication mechanisms, i.e. FIPA-standard [5]. They can also communicate this way with external systems, e.g. operator displays, MES (manufacturing execution system), ERP (enterprise resource planning) and remote monitoring systems, provided that these also have agent communication capabilities. The agents form a hierarchy based on authority relations. The leaf agents supervise some part of the controlled process and its related automation system as their areas of responsibility. The areas of responsibility may map either to the physical or functional division of the process. Higher-level agents supervise larger subprocesses via their subordinates.

The subprocess agents cooperate via both vertical and horizontal channels. The purpose of the vertical channel is to decompose operations into parts, like control actions on a certain subprocess. The horizontal channel is for resolving possible conflicts and synchronizing the suboperations, e.g. checking a condition on a process measurement before a certain action may be performed. Each agent performs its operations combining local actions and cooperation with other agents via both of these axes. How the cooperation actually takes place depends on the type of operation as will be explained in chapter 4.

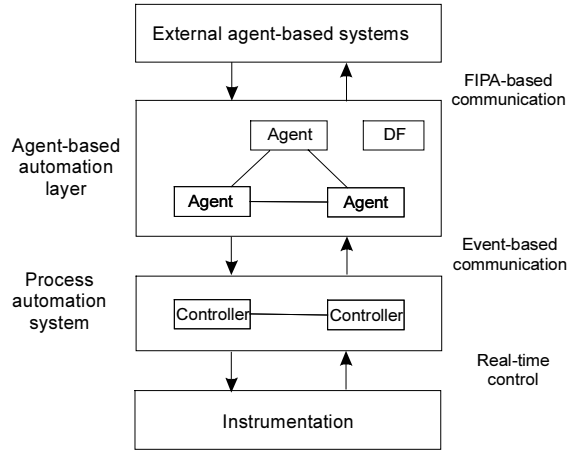


Fig. 1. Architecture of a process automation system extended with subprocess agents. Acronym DF denotes Directory Facilitator defined in FIPA.

The subprocess agents are all based on similar architecture as illustrated in Fig. 2. A subprocess agent has communication capabilities for exchanging messages with the underlying automation system and other agents. It also has data models of process instrumentation and the agent society. For its main operations the agent has modules for cooperative situation monitoring, action planning, plan execution and query processing. The action planning can be performed both in a deliberative or reactive fashion. The agent can be configured for a specific process and for a particular application with a set of configuration, rule and plan files. The overall operation of an

agent is managed by an underlying agent platform, which runs agents as multi-threaded applications.

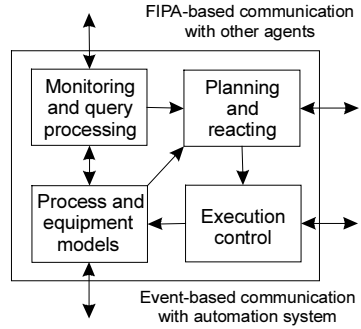


Fig. 2. Architecture of a subprocess agent.

4 Cooperation Methods for Subprocess Agents

The subprocess agents need several different MAS-based cooperation mechanisms in order to be able to perform their tasks. This is due to the different characteristics of the tasks agents need to perform. Currently, we have specified cooperation models for distributed planning and execution of goal-oriented control sequences, market-based negotiation in controller tuning and cooperative query processing for external systems. However, this list of cooperation models is not intended to be exhaustive. Particularly cooperation models for monitoring and diagnostics need to be added.

The subprocess agents create plans of control sequences during run-time in order to reach given goals concerning the status of the controlled process, e.g. *startup* or *fill tank*. The planning process can be characterized as distributed and cooperative planning. Each agent creates locally its part of the overall plan and adapts it to other agents' plans via a. FIPA Contract Net [5], [19] based negotiation process.

The negotiation process is carried out via both the vertical and horizontal cooperation channels. On one hand, supervisor agents assign subgoals to their subordinates, e.g. *startup* of the tank is a subgoal of the *startup* of the whole process. On the other hand, peer agents request goals from their peers in order to handle interrelations between their plans, e.g. tank has to be filled before pump is started. The local planning processes of the agents create plans that fulfill goals originating from both of these negotiations. The plans can contain both local actions and goals to be assigned to subordinate or peer agents. For each of the assigned goals a separate negotiation process is initiated. At first, successful negotiations lead to tentative contracts with other agents. Later, if the agent who started the planning process observes that it can fulfill all of its goals, it commits to its contracts and passes the commitments to other agents with accept messages. The distributed planning process between the subprocess agents is illustrated in Fig. 3 and described with more details in another publication [18].

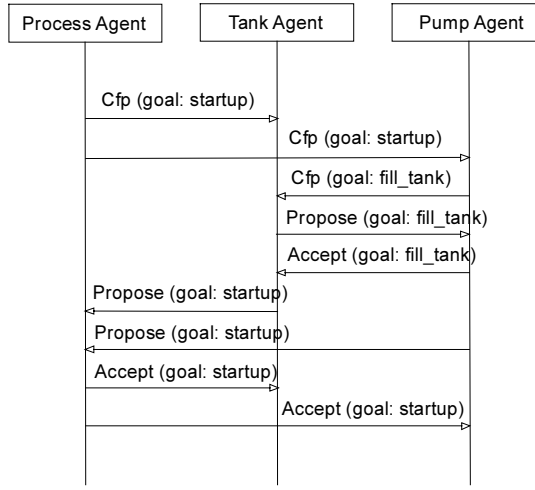


Fig. 3. Example of the distributed planning process among the process automation agents.

During the execution process subprocess agents execute those plans, to which they have committed in the planning process, in a distributed and a cooperative manner. After planning each agent has plans and contracts that specify the needed process control and communication actions. Process control actions are executed locally, while communication actions are used to coordinate and schedule actions between the agents. An agent can request its subordinate or peer agents to execute plans in order to achieve a certain process state related goal. The executing agent informs the requesting agent when this goal is achieved. During the execution process agents use FIPA Request interaction protocol as illustrated in Fig. 4.

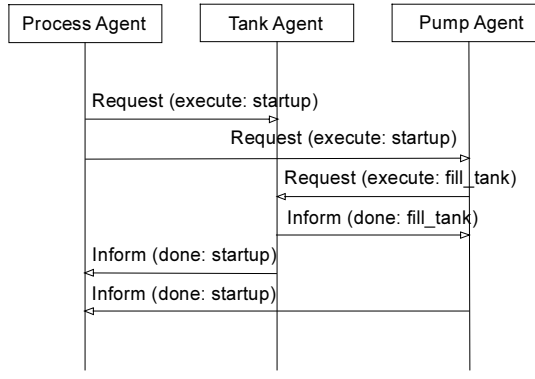


Fig. 4. Example of the distributed execution process of a process control plan.

Market-based negotiation can be used in situations where subprocess agents have competing goals that need to balanced, e.g. when tuning several controllers. The cooperation protocol in our approach to market-based negotiation is the FIPA Contract Net as illustrated in Fig. 4. First, the agent who observes a need for adjusting

a control parameter can check if it can do it with those actuators it controls itself. In addition to this, it can initiate bargaining with those subprocess agents that have registered a capability to somehow affect the control parameter. The negotiation can take place both among peers and between a supervisor and subordinate agents. Each agent can assign a cost to its operations depending on its control objectives. Based on the received proposals the initiator agent can calculate to which extent it should accept help from other agents and to which extent it should rely on its own control actions in order to minimize the total cost. Finally, all the agents carry out the agreed control actions in a parallel manner. The negotiation process may also be repeated iteratively if an acceptable result is not gained at once.

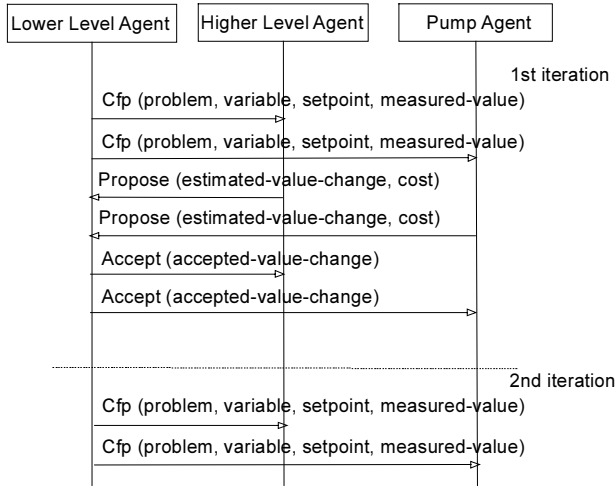


Fig. 5. Example of the iterative market-based negotiation process between the subprocess agents.

In query processing the hierarchical organization of the subprocess agent society can be utilized. Several FIPA interaction protocol may also be used. The FIPA Query interaction protocol can be used to query information from an agent, e.g. about the status of the controlled process or the underlying automation system. A query can be started by sending a message, e.g. to a supervisor agent. The supervisor agents decompose queries to their subordinates while leaf agents retrieve the actual data from the automation system. The query result is formed in the opposite direction. The FIPA Subscribe interaction protocol can be used for being informed about changes in the controlled process or the automation system. The subscription message is decomposed similarly to the query message using the subprocess agent hierarchy. The logical condition describing the subscription is stored in the agents that evaluate it periodically. Possible changes are then communicated upwards in the agent hierarchy or they can be sent directly to the client. The utilization of the hierarchy in query processing can also be combined with usage of a directory facilitator defined in FIPA.

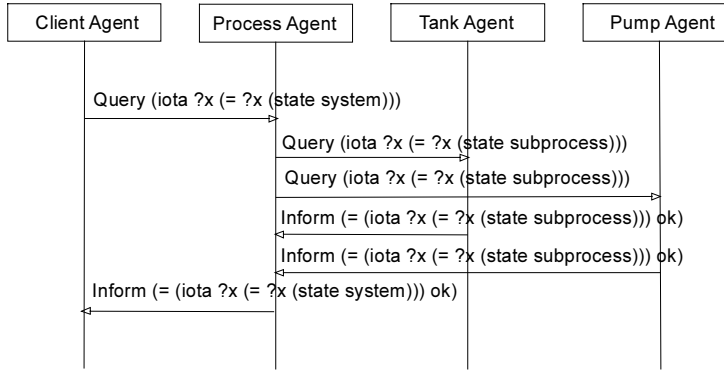


Fig. 6. Example of the cooperative hierarchical query processing among the subprocess agents.

5 Test Scenarios

A laboratory test environment has been used to experiment with the subprocess agents based process automation approach in order to test the cooperation methods. The test environment consists of a test process, an automation system and a prototype agent layer implementation [18], [17].

The test process is a small-scale water temperature control process illustrated in Fig. 7. The water level and temperature in the tank are controlled with five control valves. Process flow is imitated by circulating water with the help of a pump. The instrumentation of the process also includes several temperature sensors and a pressure and a flow sensor. At the moment the control system runs three control loops: one for water level control and two for temperature control at the upper and lower levels of the tank. The automation system of the test environment utilizes Foundation Fieldbus technology [4]. The agent layer implemented with open source software tools FIPA-OS [6], JAM [8] and Jess [10] runs on a PC with an OPC connection [14] to the automation system.

Two different scenarios, process startup and fault-recovery, with different cooperation and decision-making mechanisms have been studied with the test environment. A prototype agent layer implementation consisting of five agents (see Fig. 7) were used in these experiments. For the experiments the agents were configured by defining their areas of responsibility, rule bases and plans.

The goal of the process startup scenario is to run the process from a shutdown state into a normal operation state. The task of the agents is to create a feasible and synchronized shared startup sequence for the current initial state using their local startup plans and then run the sequence. This scenario demonstrates the cooperation in distributed planning and execution of goal-oriented control sequences.

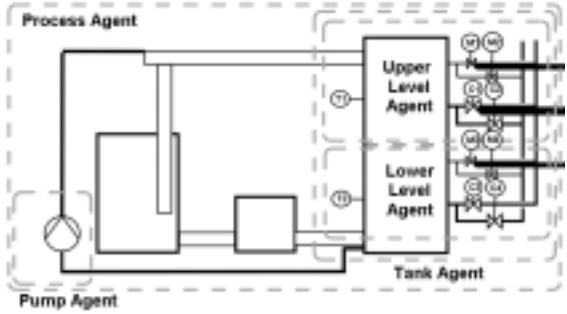


Fig. 7. Areas of responsibility for the subprocess agents supervising the test process.

The planning process starts from the Process Agent, which passes a *startup* goal to its direct subordinates, the Pump Agent and the Tank Agent. The Pump Agent has a startup plan with an initial condition requiring the tank to be full before water circulation is turned on. So it passes a goal *fill tank* to the Tank Agent. This agent uses its startup plan illustrated in Fig. 8, agrees to fill the tank and passes the *startup* goal to its subordinates. The planning process is repeated similarly between the Tank Agent and its subordinates. At the end of the planning process both the Tank Agent and the Pump Agent pass their commitment to the process *startup* goals to the Process Agent. After successful planning the Process Agent can request plan execution.

```
Plan: {
NAME: "Startup tank subprocess"
GOAL: ACHIEVE startup;
BODY:
  ACHIEVE tank_filled;
  ACHIEVE water_level_control_on;
  EXECUTE RunNegotiation.execute "upper_level" "startup";
  EXECUTE RunNegotiation.execute "lower_level" "startup";
}
```

Fig. 8. Simplified version of the process startup plan template of the Tank Agent.

The objective of the fault recovery scenario is to partially compensate the effects of a fault in the automation system. The task of the agents is to find out new setpoints to the control vales and the pump so that the influence of one broken control valve is balanced between both controlled temperatures. This scenario demonstrates market-based negotiation in controller tuning.

The three subprocess agents who react to the fault situation with the market-based negotiation scheme are the Lower and Higher Level Agents and the Pump Agent. First, the Lower Level Agent detects the fault, makes an inference that it cannot control the temperature T2 (see Fig. 7 and Fig. 9) any more and requests help from the other two agents with a call-for-proposal message describing the problem. Both the Higher Level Agent and the Pump Agent offer to change their set-points in order to reduce the problem. The Lower Level Agent accepts both proposals and also

decides to close its other control valve and start its magnetic valves, M1 and M2. These are reserve valves that are not used for control, but which can be utilized e.g. in fault situations. The negotiation process is repeated until the control error of T2 decreases below suitable limits (see Fig. 9).

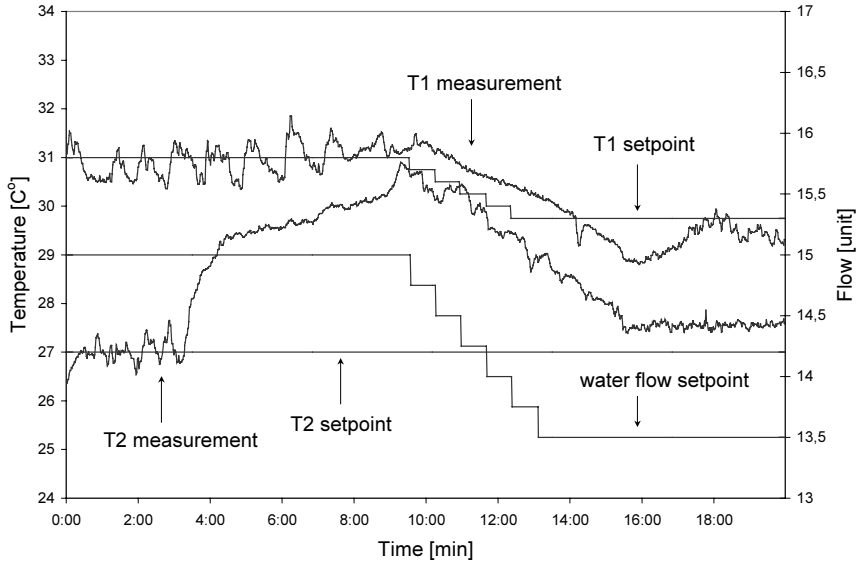


Fig. 9. Example test run with iterative negotiation. The setpoints of the temperature T1 and the water flow are adjusted iteratively.

6 Conclusions

In this paper we have presented an approach to extend a process automation system with cooperative subprocess agents. The approach is based on a subprocess agent layer, which acts on top of an ordinary automation system. The operation of this layer is designed with the techniques of cooperative MAS. The agents utilize several different cooperation mechanisms via both vertical and horizontal coordination axes. However, the current set of cooperation mechanisms is still not complete and need to be extended, e.g. with appropriate methods for cooperation in monitoring and diagnostics.

Acknowledgements. The authors would like to thank Mr Shanku Chakraborty and Mr Milan Fajt for their valuable contribution to the implementation of this study.

References

1. Chockshi, N. N., McFarlane, D. C.: Rationales for Holonic applications in chemical process industry. In: Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.): Multi-Agent Systems and Applications II. Springer-Verlag, Germany (2002) 323–335

2. Cockburn, D., Jennings, N. R.: ARCHON: A distributed artificial intelligence system for industrial applications. In: O'Hare, G. M. P., Jennings, N. R. (eds): Foundations of Distributed Artificial Intelligence. Wiley & Sons (1995)
3. Ferber, J: Multi-agent systems: An introduction to distributed artificial intelligence. Addison-Wesley (1999)
4. Fieldbus Foundation: http://www.fieldbus.org
5. FIPA: http://www.fipa.org
6. FIPA-OS: http://fipa-os.sourceforge.net
7. Hayzelden, A. L. G., Bourne, R. A. (eds.): Agent technology for communication infrastructures. John Wiley & Sons (2001)
8. JAM: http://marcush.net/IRS
9. Jennings, N. R.: An agent-based approach for building complex software systems. Communications of the ACM, Vol. 44, No. 4. 35–41
10. Jess: http://herzberg.ca.sandia.gov/jess
11. Kuikka. S.: A batch process management framework, Domain-specific, design pattern and software component based approach. VTT Publications No. 398. Espoo, Finland (1999)
12. Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.): Multi-agent systems and applications. Springer-Verlag (2002)
13. Maturana, F., Tichy, P., Staron, R., Slechta, P.: Using dynamically created decision-making organizations (holarchies) to plan, commit and execute control tasks in a chilled water system, In: Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA'02) (2000)
14. OPC Foundation: http://www.opcfoundation.org
15. Rijnsdorp, J. E.: Integrated process control and automation. Process measurement and control, 2. Elsevier Science Publishers (1991)
16. Sanz, R.: Agents for complex control systems. In: Samad, T., Weyrauch, J. (eds.): Automation, control and complexity. Wiley & Sons, England (2000) 171–190
17. Seilonen, I., Appelqvist, P., Halme, A., Koskinen, K.: Agent-based approach to fault-tolerance in process automation systems, In: Proceedings of the 3rd International Symposium on Robotics and Automation (ISRA'2002). Toluca, Mexico (2002)
18. Seilonen, I., Pirttioja, T., Appelqvist, P., Halme, A., Koskinen, K.: Distributed Planning Agents for Intelligent Process Automation, Accepted to the 5th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2003). Kobe, Japan (2003)
19. Smith, R. G.: The Contract Net protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computers, Vol. 29, No. 12 (1980) 1104–1113
20. Tichy, P., Slechta, P., Maturana, F., Balasubramanian, S.: Industrial MAS for planning and control. In: Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.): Multi-Agent Systems and Applications II. Springer-Verlag, Germany (2002) 280–295
21. Tommila, T., Ventä, O., Koskinen, K.: Next generation industrial automation – needs and opportunities. Automation Technology Review 2001, VTT Automation (2001) 34–41
22. Wang, H., Wang, C.: Intelligent agents in the nuclear industry. IEEE Computer, Vol. 30, No. 11 (1997) 28–34
23. Weiss, G. (ed.): Multiagent systems. MIT Press (1999)
24. Ygge, F., Akkermans, H.: Decentralized markets versus central control: A comparative study, Jornal of Artificial Intelligence Research, Vol. 11. (1999) 301–333

Evaluating a Holonic Packing Cell

Martyn Fletcher[1], Duncan McFarlane[2], Alan Thorne[2], Dennis Jarvis[1], and Andrew Lucas[1]

[1]Agent Oriented Software Ltd. (AOS),
Mill Lane, Cambridge CB2 1RX, United Kingdom.
{martyn.fletcher,dennis.jarvis,
andrew.lucas}@agent-software.co.uk

[2]Institute for Manufacturing, University of Cambridge,
Mill Lane, Cambridge CB2 1RX, United Kingdom.
{dcm, ajt}@eng.cam.ac.uk

Abstract. Nowadays there is a proliferation of research into multi-agent and holonic systems. These systems are being applied to environments including production, supply chain and warehousing to increase the flexibility, openness and mass-customisation of e-manufacturing operations. One such example is the Holonic Packing Cell demonstrator at Cambridge University's Institute for Manufacturing. However, there is very little basis for evaluating how well such systems have been built or how they will operate once they are deployed into pragmatic shop-floor settings. This paper is an initial step in filling this void by proposing a framework to evaluate holonic systems with respect to: (i) the performance of their controlled operations, (ii) the applicability of the software and control systems, and (iii) the methodology used.

1 Introduction

Over the past decade, a number of academic and industrial research organisations have been investigating the issues associated with the development of holonic systems and how they can be effectively deployed into manufacturing and supply chain management environments [1-3]. The driving forces behind these studies are mainly commercial [4]. Market forces are pushing manufacturing businesses to provide mass-customisation of their product families and react more quickly to consumer demands. Meanwhile, these companies do not wish to discard their existing investment in hardware or brand marketing. Hence a new technological approach is needed to make, handle and transport products in a flexible manner to cope with these 'short production run' demands in a 'business as usual' way [5]. There are very few holonic systems deployed in factories making real products. One of the main reasons for this lack of uptake [14] is that a number of research issues remain that have yet to be resolved. One such issue is how to evaluate the design, construction and operation of a holonic system. As engineers, we believe that the holonic system that we will describe in the paper is better, as judged on several criteria, than other systems. Yet at present there is no coherent mechanism to evaluate a holonic system. This paper is an initial step in addressing this topic, and presents an evaluation framework for holonic

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 246–257, 2003.

systems' performance, applicability and methodology. To highlight some aspects of the framework and how it can be used to assess holonic systems, we refer to a holonic packing cell implemented at Cambridge University's Institute for Manufacturing.

2 A Worked Example: The Cambridge Holonic Packing Cell

A driving force behind the construction of the Cambridge holonic packing cell has been the requirement to have a test bed upon which experiments relating to agile and intelligent manufacturing [15] can be conducted and measured. This test bed now provides us with the opportunity to evaluate different design, development and integration strategies with a view towards formulating a methodology that can be used repeatedly to build high-performance holonic systems [16]. The layout of the cell is illustrated in Fig. 1.

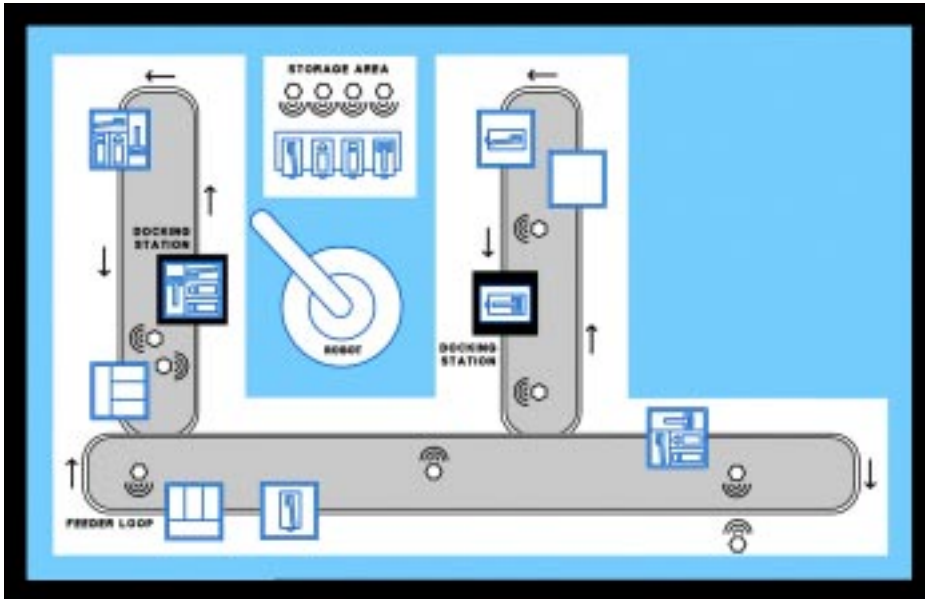


Fig. 1. Layout of the Cambridge holonic packing cell.

Physically, the packing cell has three conveyor loops to transport batches of Gillette™ personal grooming products (razors, shaving gel, deodorant and shaving foam) and gift boxes into which these items will be packed. This Montech track has independently-controlled gates that navigate shuttles into and out of loops. There are also two docking stations where shuttles (carrying boxes) are held so that a Fanuc M6i anthropomorphic robot can pick and place items into the boxes. The system also comprises a storage unit to hold items using a first-come-first-served discipline. Sensors and actuators within the cell (apart from the Fanuc robot) are connected to an Omron SYSMAC CV500-V1 Programmable Controller, which in turn is attached to a

Personal Computer via Ethernet, in order to support a blackboard. This blackboard maintains a copy of the data registers held by the controller relating to the status of sensors and actuator parameters that the control system can read from and write to in order to get the physical hardware to perform desired actions. The aim of the packing cell, as a whole, is to pack items into boxes in a flexible way to meet specific retail needs that vary over time. The system (see Fig.2) is an example of mass-customisation by letting the customer select (via a web interface) any three of the four Gillette product item types, and pack them into one of two box styles.

Fig. 2. The holonic cell packing Gillette items after an order has been issued over the Internet.

Holons have been introduced to better utilise factory resources, make products more efficiently and thereby improve the reaction of manufacturing businesses to changing market requirements. For our purposes, we define a holon [17] as containing either physical hardware or some information services coupled with an intelligent agent [18,19]. A holon can then provide both autonomous actions and support cooperative interactions. The agent elements of the holons in the cell have been encoded using JACK Intelligent Agents™ platform from Agent Oriented Software Limited.

There are the following *resource holon* classes in the system, with the number of instances shown in brackets: Robot (1), Docking Station (2), Gate (2), Reader (7), Track (1), Box Manager (1), Storage (1), and Production Manager (1). There are also *order holons*, with one being spawned for each gift box, in order to orchestrate how that box must be manufactured via collaboration with the various resource holons.

Manufacturing scenarios that the cell, and its intelligent holons, can demonstrate are:

- *Batch Orders*: Introduce batch orders so individual orders manage their packing (e.g. acquire a suitable box, shuttle and items). Also included is negotiating with resources (including docking stations and the robot) to achieve processing goals.
- *Unpacking*: Unpack completed boxes so urgent orders can be satisfied quickly.

- *Reconfigure Docking Station Processing*: Disable a docking station (i.e. mimic a failure) and so cause the holons to reconfigure themselves to process work elsewhere. The holons can also handle rush orders that must be packed quickly.
- *Storage Handling*: Handle storage of item in a reactive manner by bringing new shipments into the cell, hold them in storage chutes and handle them correctly.

Using a methodology to introduce holons into such a cell may have important benefits in terms of development costs, flexibility and robustness of the resulting system. Yet there may be drawbacks in comparison to traditional approaches, for example the performance of the resulting system may not be as good as a system dedicated to low-variety high-volume production. Therefore a framework to evaluate and compare holonic systems is clearly needed.

3 Framework for Evaluating Holonic Systems

Clearly the desired responsive behaviour of an enterprise that supports mass-customisation can be viewed from two perspectives. For the client, it is to pack boxes to meet their requirements and do so promptly within a budget that they set. For the manufacturing business, it is to maximise the cell's resource utilisation and hence make a profit. For the adoption of holons within the enterprise to be considered a success, we need to show that the added responsiveness that the system offers is achieved in a cost effective and practical manner. There is little benefit in having a flexible factory if the algorithms that holons must run to make products are so slow that the throughput is merely a fraction of what it would be if a mass production model was in place. As part of a quantitative evaluation, criteria such as performance must be measurable to asses how well the system is operating, yet many criteria cannot be easily measured on some scale and so qualitative evaluations must suffice. Both qualitative and quantitative measures are incorporated within our evaluation framework to help judge the relative merits of some holonic feature.

For example, if the customer's perspective is focused on having their orders fulfilled by requested deadlines (rather than on cost or quality say) then the cell's throughput may be analysed by computing the mean time between an order entering the system and shipment of all boxes out of the factory. Alternatively from the business's viewpoint, other sophisticated measures can be constructed to take into account how the throughout of orders with differing levels of urgency or pay-off is managed. At present in our holonic packing cell, the customer sets a single price per box for the entire batch. Yet in subsequent phases of the cell's implementation, it is assumed that there is some measure of financial cost associated with both the time taken for certain boxes to be shipped and the value of items in those boxes. This measure of cost is linked to the client's willingness to pay more for quicker delivery of varying amounts of high-value or scarce goods.

To evaluate how well a holonic system operates in a scientific and repeatable manner, we need a framework. This framework is characterised by the following guidelines:

- Develop manufacturing systems that handle a dynamically changing environment without having to centralise all the control.
- Keep knowledge on what the product (e.g. a packed box) should 'look like' separate from the machine instructions used to achieve these features (ie the actions of the docking station, robot and so forth to pack items into the box). This improves the potential for allocating a packing job to several heterogeneous cells.

The evaluation framework's central proposition is the creation of a uniform model for assessing and judging the relative merits of holonic systems' design and operation. This goal is underpinned by two inter-related factors:

- The need to overcome the fragmented perspectives of experts who are working on various aspects of the holonic system research spectrum. Lacking a single vision of how to assess a holonic system (i.e. not having a level playing field) can distort experimental practice and may introduce duplications of effort, which could continue to hamper the quick deployment of holons into today's industry.
- The design and operational inefficiencies in many holonic concepts have given rise to a range of problems and can adversely impact the quality of the resultant manufacturing control system. These problems mean that it is often unclear how the merits of one concept in one system relate to other ideas or to the same principle as adopted in other systems.

To satisfy these goals a unified evaluation framework is required. Our proposed evaluation framework uses a *spidergram.* This graphically representation helps us to assess the design and operation of one or more holonic systems. The inspiration for this paper is Miles and Baldwin's article [23] where quantitative and qualitative criteria, such as clear-up rates and crime prevention activities, of 50 UK Police Forces were mapped onto spidergrams to display and contrast their relative performance. A good overview of how to represent complex data using spidergrams is given in [22]. Our holonic system evaluation spidergram shows percentages along each edge so that a perfect holonic system would score 100% based on some quantification.

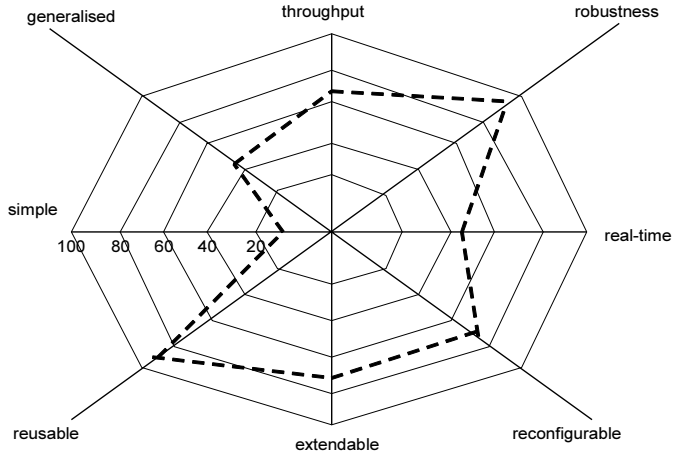


Fig. 3. Spidergram to illustrate the evaluation of a holonic system from multiple viewpoints.

A spidergram has been selected as a suitable presentation format because it has facilities, based on relative performance of observable measures, to: (i) contrast different facets of a single system, and (ii) compare multiple implemented holonic systems. The authors reviewed several schemes to evaluate holonic systems including reviews of a person's employment [25] or approaches similar to how a reviewer may asses the merits of an EU project proposal [26]. These techniques have their merits including being to measure advances in a person's employment skills or estimate a project's outcomes respectively. Progress in these areas is gauged by experienced people carrying out specific tests, manipulating the data using weights and thresholds to form a single score, and judging this result against established benchmarks. They also have disadvantages, e.g. an assesment's stakeholder can misinterpret a single figure because they do not view all information in the proper context.

In terms of our packing cell, stakeholders such as production managers, academics and decision-makers in manufacturing businesses may fail to take into account factors such as how the system is very good in some respects but is poor in three other areas. They would also fail to take into account the reasons why the results were obtained such as the holons' diverse architectures and capacity for intelligent reasoning. Hence we argue that a balanced judgement cannot be made using such solitary scores. Therefore the authors selected spidergrams as a suitable scheme to evaluate a holonic system, noting that spidergrams could be used in conjunction with other graphical and statistical models depending on the nature of the evaluation to be undertaken. One of the merits of using a spidergram is that by computing the area occupied by a holonic system's spidergram, the diagram provides an overall system score. This number can then be used to rank various holonic designs or implemented systems, and carry out competitor analysis. For example, the spidergram in Figure 3 shows an evaluation of our holonic packing cell with good robustness and reusability but poor simplicity.

The clockwise sequence of metrics is not fixed and can be arranged to graphically illustrate the features of holonic system to reflect the evaluator's opinion. This lack of restriction can, of course, be viewed as a weakness or a strength of the framework. For instance, arranging a spidergram's metrics to alternatively display high and low score gives the impression of a 'star' that may be visually less criticised than a cam-shaped spidergram when the metrics are re-sequenced. The number of metrics can be increased and the percentages altered to reflect a revised quantitative assessment strategy and so the evaluation framework is sufficiently flexible to cope with any holonic system and any set of benchmarks. The metrics to measure a holonic system can be categorised into three evaluation groups: (i) *performance of controlled operations* (throughput and robustness); (ii) *applicability of software / control system* (real-time, reconfigurable, extendable); and (iii) *methodology used* (reusable, simple, generalised). The groups are discussed in subsequent sections of this document. There are no strong rules for placing metrics into particular groups, nor are there rigid guidelines for arranging groups in any given sequence clockwise around the spidergram. This lack of guidance and being deficient in stringent techniques to generate a quantitative value for how well a holonic system performs in some metric are key drawbacks to the evaluation framework. These are topics for further research.

4 Evaluation Group 1: Performance of Controlled Operations

This group measures the performance of the controlled operations in the packing cell with regard to the overall throughput of goods in some time period, and how robust the holons are to changes of how, where and when packing operations are performed. This group is geared towards satisfying the commercial goals of the manufacturing business. Yet a customer-oriented perspective is of equal benefit because we might want to evaluate other metrics depending on the customer, production environment and so on. For example, the holons may need to reconfigure themselves in order to minimise the average lateness so that the highest degree of customer satisfaction is achieved. Here we concentrate on robustness because a system that is reliable is of key importance to factory managers, since a major factor impinging on manufacturing volume is the mean time between failures and the limited opportunities a control system has to recover. A primary goal of holonics is to alleviate this problem.

To best ensure efficiency in the case of failures, holons may need to perform some degree of contingency planning. This is a complex task. It includes taking into account task deadlines, the inter-connectivity between subtasks, availability of other resource holons that could adopt the role of executing that task, and other criteria imposed by the order holon (e.g. overall completion cost for the task, quality of service requirements etc.). One option is for the cell, via all of the holons interacting among themselves, to generate and execute a high-level control policy that determines the best course of action for every resource holon to achieve the cell's production goals in terms of optimising business criteria and resource allocation parameters. Yet the resource holons in our current cell do not generate or maintain such meta-level policies due to the communicational overheads. Hence we argue that individual holons should reliably execute their autonomous actions so global reliability emerges.

A key advantage of using JACK is that the agent-based holons are constructed using a solid Belief Desire Intention (BDI) execution engine. This helps model the holon's desires in terms of goals, binds these goals to plans at runtime, and can easily handle failure of agents' plans by re-issuing the goal and binding a different plan to it. Let us consider a theoretical example: the order holon's recipe to pack a given box specifies that the box should be processed on a docking station within a fixed deadline. If the nominated docking station fails then the station holon informs the order holon. The order holon then creates a set of alternative stations and times that satisfy this high-level goal. The best candidates are determined using some statistical evaluation of the recipe's criteria, e.g. candidate docking stations offering early time slots may be preferred. In terms of our evaluation framework, we argue that a holonic system is awarded points if it satisfies certain robustness criteria and looses points if it performs badly. For instance, if a customer-oriented approach is used then the system deserves up to 30% higher than system solely using a business-oriented approach. If contingency planning and a solid BDI model are used during implementation then the system should be rewarded up to 20% and 25% respectively. If re-assignment of work is done using a dynamic set of candidates then this should be reflected with an assessment of up to 25%. This list of ways to convert qualitatively arguments into percentages is not exhaustive, and the percentages are open to revision.

5 Evaluation Group 2: Applicability of Software/Control System

The applicability of a software / control system group examines how well holons handle real-time decision-making as well as their agent-based deliberation and collaboration techniques. Real-time decision-making is critical in dynamic environments where it is not appropriate to perform a fixed sequence of predetermined actions. Therefore a system's ability to make real-time decisions based on sensing the environmental state is an important factor in whether or not holons can successfully be applied in a particular manufacturing situation. The group also measures how reconfigurable the holons are, in terms of having their control structures and decision-making processes changed at runtime and being re-deployed into different operational circumstances with limited 'swap over' time. Again the extent of the support for run-time reconfiguration is a key factor in determining the range of applications that the system can be put to. The third metric in this group is how extendable the holons. Extensibility relates to how well holons cope with new knowledge and new autonomous skills, making new products with the existing hardware and cooperating with new holons. Since, an extendible holon system can be readily expanded to cope with additional robots and other manufacturing resources, extendibility tends to lead to systems with greater applicability. The focus is on extendibility, because this metric is critical if holons are to be introduced seamlessly into factories and interact effectively with legacy systems such as Manufacturing Execution Systems (MES). It should be clearly noted that the approach described below is used just to illustrate one way to get applicability from the holonic system.

In the Cambridge packing cell, neither rigid nor hierarchical inter-holon organisations are defined. Each holon begins with some knowledge concerning which other holons (peers) it should talk in order to achieve some task. This talking is implemented using message events that are passed between agents. An arguably better approach is to replace holons implemented with autonomous agents with an implementation using JACK Teams™ [27] because team-oriented computing provides a rich set of constructs and would support multi-holon cooperation through holarchies. A paper explaining the relationship between JACK Teams™ and holonic principles is being written [24]. With JACK Teams™, interaction among holons is via built-in constructs (roles, named roles, teamplans and teams). These constructs are used to encapsulate the exchange of task requests and knowledge, and provide a clean interface between holons without having to pre-define which holon instance takes on a particular job at execution time. This is as extensible approach that allows reconfiguration at runtime. For example, it can be used to ensure effective and fair allocations of resource holons' valuable commodities (e.g. time slots) across the order holons that need them.

Another argument in favour of using roles as a means of increasing extensibility is that social relationships among holons built upon roles provide a conceptual framework in which each holon either: (i) plays its role as a resource producer or resource consumer, or (ii) occupies a well-defined position in the society. For example, the Fanuc M6i robot holon may take on the role of material handler inside the cell, yet this role may later be filled by a Fanuc A520 or Mitsubishi RV-1A/2AJ robot – the identified roles are static but the specific holon occupying the role is

dynamic. Moreover, a production manager holon or a track holon occupies a critical position in the organisation to orchestrate the sequence order holons are spawned or monitor where shuttles are currently located. These knowledge server types of holon roles can be taken only by a specific class of software holon, but they can be re-started at any time if failure occurs. Such a role-based approach allows us to:

- *Re-structure and revise holon interactions*. This is within the control system, and between the control system and existing business information systems.
- *Re-assign capabilities.* Encapsulate knowledge and functionality within different holons without having the stop execution, edit, re-compile and run the software.

The role that a holon adopts determines how extensible its is in terms of:

- ***Skills.*** Skills represent the functions, services and knowledge processing needed by the holon to assume the desired role. Each resource holon is characterised by the set of abilities that are needed to manage that hardware. For example, an extendible docking station holon should have the facilities to: (i) determine if and when it should process a shuttle, (ii) interact with the hardware to grab and release shuttles, and (iii) cooperate with order holons to allocate timeslots in the schedule and inform them once jobs are completed. Extendible order holons are characterised by the range of skills they need to get themselves manufactured. For example, cooperating with resource holons (according to a production recipe) and cooperating with other orders to buy and sell their partially-completed boxes so urgent orders can be delivered on time. By designing the resource and order holons' skills in a sufficiently general way, the system is made more extensible.
- ***Responsibilities.*** When a holon assumes a role at execution time, the holon becomes responsible for the successful completion of the tasks associated with that role. For example, when the Fanuc M6i robot holon accepts the role of material handler it becomes obliged to pack boxes, unpack boxes, or sort the storage chutes until that role gets re-assigned. One mechanism to ensure that a responsibility is successfully completed is to issue request messages to the resource holons for each goal in the order holon's recipe. If a resource holon becomes unable to complete its responsibilities during execution of the job, it informs the order holon, which is then required to find another resource that can offer the affected role and negotiate with that holon to maintain the processing.
- ***Knowledge Sharing***. Data is requested and provided between order and resource holons so that the resource holon assuming a role can fulfil its responsibilities. This exchange is achieved, over Ethernet with guaranteed delivery, by attaching strictly-typed data and objects onto the messages that are passed among holons.

When comparing two holonic systems, then we would expect the role-based system to achieve better scores along the extendibility metric. Therefore the packing cell deserves a relatively high extendibility score (at least 50%) because it uses simple resource request/response messages within roles and it enable resource holons to dynamic adopt their responsibilities. The packing cell is not idle because it does not let new resource holons be added dynamically to fill each role, or let the system expand at runtime without the need for re-engineering.

6 Evaluation Group 3: Methodology Used

This group evaluates the methodology utilised to develop the packing cell in terms of how reusable, simple, and generalised this philosophy and its methods are. Simplicity in a methodology results in implemented holons that have both simple interactions and autonomous decision-making. To achieve such simplicity, the methodology supports the decomposition of the control system into well-defined units and provides design guidelines. These guidelines are often the first concern in modern software engineering models, like object-oriented analysis, and the development of holonic systems is no exception. However, the main difference is the dynamism that a holonic system exhibits. So the methodology's guidelines should compensate for this dynamic behaviour, and be easy to comprehend. We propose that a suitable methodology will use a spiral approach with each cycle having the subsequent phases and models:

- ***System Design Model.*** An anthropomorphic model of the holonic system's requirements and purpose is established. This leads to identifying holon types, their private actions, interactions and the interfaces to the legacy factory and business systems. Developing this model involves creating a functional description of the system using UML concepts, and exploring each holon's responsibilities and collaboration metaphors through role-specific scenarios.
- ***Holonic Society Model.*** An infrastructure for the interactions and dependencies among holons is then constructed via two steps. First, holon roles are described using class diagrams to isolate distinct roles adopted by holon types, the tasks involved when a holon takes on that role and the communication metaphors used to accomplish the role. Second, the conversation protocols, grammar and pragmatic knowledge structures used by holons are designed.
- ***Implementation Model.*** Holons are encoded as a solution architecture using the JACK Agent Language constructs (i.e. agents, capabilities, events, plans, and belief structures). Source Java code is produced and executed on a suitable platform using the Java virtual machine and the JACK runtime libraries.
- ***Test and Refinement Model.*** Individual holon, multi-holon interactions in the society and overall holonic system behaviours are verified against requirements. If problem solving is not validated then refinements are identified for next cycle.

By applying this methodology, we make the following proposition: a simple methodology constructs an efficient holon society so that each holon acts and interacts in a comprehendible manner, and that can be easily changed or reconfigured. The methodology used to build the packing cell did not satisfy this proposition and so it deserves a relatively low score on the simplicity metric.

7 Concluding Remarks

From a software engineering perspective, this paper has presented research on constructing a framework to evaluate the design and operation of holonic systems. Compared with current literature [6-12], the paper's work has some unique features.

- *Application-Oriented Features***.** The work in this paper is geared towards direct applicability in real production and logistic environments. The mechanisms in [6] and [7] also present approaches to developing a methodology for constructing

holonic systems, but the main purpose of that formalism is not for creating real holonic implementations. Rather, it is geared to providing a migration strategy to move from traditional hierarchical control towards a potential framework for holonic control based upon aggregating components in a bottom-up manner. An orthogonal (top-down) approach has been described in [8] where manufacturing orders are decomposed with the aid of production knowledge and then jobs are dynamically assigned to machine resources.

– *Wide Applicability***.** The models presented in [9-11] cover only a small range of issues with regard to measuring a control system's operational performance, and each does so from a rather narrow perspective. The flexibility of the approach outlined in this paper implies that it has a wide applicability across a number of agile manufacturing and supply chain domains. Unlike the agent-based models proposed in [12] and [13], this paper's spidergram model is designed to support a plethora of metrics (for re-usability etc.) that can express the system's operations. It can be directly used to integrate various evaluation criteria, in the form of *metric groups*, for judging the relative merits of one design against another. Even the types and number of instances for each holon class can be changed in order that the framework be embedded into different applications, e.g. to measure how close the integration is with legacy e-business solutions or to test how interoperable the holonic system is with Auto-ID data (an emerging standard for product tracking and identification in a supply chain) [20,21].

It could be argued that we have not made a framework to evaluate holonic systems. Rather we have postulated a quantitative spidergram and a number of qualitative metrics which we believe adequately span stakeholder viewpoints. To answer this criticism, we have tried to give some reasons why we mapped the qualitative metrics of the Cambridge cell onto the values presented in Figure 3. Another deficiency is that there is no discussion as to the adequacy of these metrics, how values are determined for the packing cell and how meaningful comparisons of these values between systems will be. To counter this argument, our future work will concentrate on collecting empirical evidence to conclusively demonstrate that the systems we build are better than legacy control systems or diverse implementations of a holonic design.

References

1. Tichy, P., Slechta, P., Maturana, F., Balasubramanian, S.: Industrial MAS for Planning and Control, Proc. of 2nd Int. Conf. on Industrial Applications of Holonic and Multi-Agent Systems, Munich, Germany, (2001)
2. Seki, T., Hasegawa, T.: IDPS Operating System for Constructing Fault-Tolerant Systems, Information Processing Society of Japan Magazine, vol 36 no 08-018, (2003)
3. van Leeuwen, E.H. (ed.): Track on Multi-Agent and Holonic Manufacturing Systems at the 5th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, Cancun, Mexico, (2002)
4. Schaeffer, C.: Holonic Production and Material Flow in Industry, Proc. of Int. Symp. on Holonic Manufacturing Systems, Kitakyushu, Japan, http://hms.ifw.uni-hannover.de (2000)
5. Sathern,. E.: Envisioning Agile Packaging – Unilever envisions new packaging machinery that treats change as business as usual by focussing on a 'holonic' approach, www.packworld.com/articles/Features/15421.html, (2002)

6. Session on Benchmarking and Performance Measures of on-line Production Scheduling Systems, at the 7th IFAC workshop on Intelligent Manufacturing Systems, Budapest, Hungary, (2003)
7. Chirn, J.L, McFarlane, D.C.: A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture, Proc. of 1st Int. Conf. on Industrial Applications of Holonic and Multi-Agent Systems, Greenwich, UK, (2000)
8. Bussmann, S., Sieverding, J.: Holonic Control of an Engine Assembly Plant – An Industrial Evaluation, Proc. of IEEE Conf. on Systems, Man and Cybernetics, Tucson, USA, (2001)
9. Van Brussel, H., Valckenaers, P., Bongaerts, L., Peters, P.: PROSA: A Reference Architecture for Holonic Manufacturing Systems, Computers in Industry, vol 31 no3, (1998)
10. Kalia, D., Khargonekar, P.P.: Formal Verification for Analysis and Design of Logic Controllers for Reconfigurable Machining Systems, IEEE transactions on Robotics and Automation, vol 18 no 4, (2002)
11. Mingsian, R.B., Kweun-Yieng, O.: Experimental Evaluation of Adaptive Predictive Control for Rotor Vibration Suppression, IEEE transactions on Control Systems Technology, vol 10 no 6, (2002)
12. Barata, J., Camarinha-Matos, L.M.: Implementing a Contract-based Multi-agent Approach for Shop Floor Agility, Proc. of 3rd Int. Workshop on Industrial Applications of Holonic and Multi-Agent Systems, IEEE, Aix en Provence, France, (2002)
13. Martin, D., Cheyer, A., and Moran, D.: The Open Agent Architecture – A Framework for Building Distributed Software Systems, Applied Artificial Intelligence, vol 13 no 1, (1999)
14. Parunak, H.: A Practitioner's Review of Industrial Agent Applications, Autonomous Agents and Multi-Agent Systems, vol 3, no 4, (2000)
15. Agile Manufacturing Benchmarking Consortium, www.ambcbenchmarking.org/, (2003)
16. Rana, O., Kotz, D.: Special Issue on High Performance Agent Systems in Concurrency and Computation: Practice and Experience, vol 13 no 1, John Wiley & Sons, (2001)
17. Marik, V., Fletcher, M., Pechoucek, M.: Holons & Agents: Recent Developments and Mutual Impacts, Multi-Agent Systems and Applications, Springer, LNAI 2322, (2002)
18. Bussmann, S., Jennings, N.R., Wooldridge, M.: Re-use of Interaction Protocols for Decision-Oriented Applications, proc. of 3rd int. workshop on Agent-Oriented Software Engineering, Bologna, Italy, (2002)
19. Wooldridge, M.: An Introduction to Multi-Agent Systems, John Wiley & Sons, (2002)
20. Heyman, D.: RFID in Action: Killer Applications for Profit, Efficiency & Economy, Proc. of 1st RFID in Action Event, London, UK, (2003)
21. Ashton, K.: Auto-ID Center: The Big Picture, Proc. of Sun / Mass e-commerce Adoption Forum, http://wwww.autoidcenter.com/media/Mass_e-commerce.pdf, (2003)
22. Lyman, T.: Adaptive Analysis of Locally Complex Systems in a Globally Complex World, Conservation Ecology, vol 3 issue 2, http://www.consecol.org/vol3/iss2/art13, (1999)
23. Miles, A., Baldwin, T.: Spidergram to Check on Police Forces, The Times newspaper, http://www.timesonline.co.uk/article/0,,2-352275,00.html, (July 10 2002)
24. Fletcher, M.: The Relationship between JACK Teams™ and Holonics (in progress), technical report of Agent Oriented Software and Cambridge University, (2003)
25. Motivational Appraisal of Personal Performance, http://www.assessment.com, (2003)
26. Guidelines on Proposal Evaluation and Selection Procedures for Framework 6 Projects, http://fp6.cordis.lu/fp6/home.cfm, (2003)
27. Jarvis, J.: JACK Intelligent Agents™ JACK Teams Manual, http://www.agent-software.com/shared/demosNdocs/teamsguide/html/, (2003)

FABMAS: An Agent-Based System for Production Control of Semiconductor Manufacturing Processes

Lars Mönch, Marcel Stehli, and Jens Zimmermann

Technical University of Ilmenau,
Institute of Information Systems,
Helmholtzplatz 3, 98684 Ilmenau, Germany
{Lars.Moench, Marcel.Stehli, Jens.Zimmermann}@tu-ilmenau.de

Abstract. In this paper, we present results on designing the architecture of an agent-based system for production control of semiconductor wafer fabrication facilities (wafer fabs). These manufacturing systems are characterized by reentrant product flows, sequence dependant setup-times, inhomogenous parallel machines, a diverse product mix, a mix of different process types, prescribed due dates of the orders and preventive maintenance due to difficult technological processes. Hence, coordination issues have to be considered during the design of a production control system for such type of manufacturing processes. We suggest a hierarchical production control scheme. We use the PROSA reference architecture in order to form the proper agency. We describe the development of a system prototype using the C++ and C# programming language and the .NET middleware. In order to allow the investigation of dynamic scenarios by emulation of the production process we suggest the use of the discrete event simulator AutoSched AP and a blackboard type data layer as a coupling component between the production control system and the simulator.

1 Introduction

Recently, the electronics industry has become the largest industry in the world. The most important area in this industry is the manufacturing of integrated circuits. In the past, sources of reducing costs were decreasing the size of the chips, increasing the wafer sizes and improving the yield, simultaneously with efforts to improve operational processes inside the wafer fabrication facilities.

Currently, it seems that the improvement of operational processes creates the best opportunity to realize the necessary cost reductions. Therefore, the development of efficient planning and control strategies is highly desirable in the semiconductor wafer fabrication domain. In the course of the development of new planning and control algorithms, the researchers and developers have to take into account the new opportunities in advanced software technologies.

Software agents allows for the implementation of distributed planning and control algorithms. The agents are able to act autonomously, on the other hand their communication abilities ensure a cooperative behavior and the fulfillment of global system

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 258–267, 2003.

goals. In this paper, we extend a system architecture for a multi-agent-system described in [6], [7] using the JAFMAS framework and RMI-based communication to the considerably more complex FABMAS system. Due to performance disadvantages of the Java programming language and RMI-based communication we decided to base the FABMAS system on the C++ and C# programming language and to use .NET as a communication infrastructure.

The paper is organized as follows. In section 2, we describe the manufacturing domain that is relevant for our research. We give a short outline to the hierarchical production control approach in section 3. Then we continue with an agentification based on the PROSA reference architecture. We describe the architecture of the resulting multi-agent-system in section 5. Section 6 provides information on the coupling of FABMAS with a discrete event simulator.

2 Production Control Problems for Wafer Fabs

In the analysis of the application domain, we have to distinguish between the physical layout of the production system, the production conditions and the control goals.

The manufacturing of integrated circuits (IC) on silicon wafers is a complex production process. Between 250-500 process steps on 50-120 different types of equipment (machines) are required to produce a medium complexity circuit. Batch processes (the processing of different lots at the same time on the same machine), sequence dependant setup times, very expensive machines and reentrant flows are typical for this type of production. The routing of lots, the dynamic entities in the system, takes place between different groups of parallel machines. A single group of parallel machines is called a work center. The work centers are aggregated into work areas. On the next level, we find the full wafer fab, i.e., the different work areas form the fab.

We find decision-making tasks on the factory level, the work area level and the work center level. We refer to [5] for a more detailed description of the semiconductor manufacturing domain and the decision-making tasks.

Because of the problem size and the complicated production conditions, including dynamic and stochastic aspects, a determination of schedules with process step granularity is difficult. Therefore, central implementations of certain dispatch strategies can be found in most wafer fabs. Because the dispatch rules are myopic with regards to time and space, coordination tasks are carried out by experienced people from the production control department or by foremen (senior operators) on the shop floor.

However, it seems to be possible to use more sophisticated algorithms and state of the art software techniques in order to find a compromise between centralized control, taking into account global aspects and fully decentralized algorithms with reactive abilities. Multi-agent systems (MAS) offer a way to obtain the desired compromise between centralized control and fully decentralized control.

3 Hierarchical Production Control Approach

Based on the described physical decomposition of a wafer fab we suggest a hierarchical multi-layer approach. We consider the full wafer fab as upper level of the hierarchy. We use a beam-search-type algorithm (cf. [3] for more details of the algorithm) for minimizing the deviation of the completion time of the lots from their planned due date. The algorithm uses an aggregated model of the process flows and capacities. We use classes from the ILOG libraries in order to implement the algorithm. As a result of the algorithm we obtain start and end dates for the lots for each single work area.

The mid layer is formed by the different work areas. We use a shifting bottleneck type algorithm (ABSBH) in order to minimize the performance measure total weighted tardiness for each single area. Here, we measure the tardiness with respect to the work area related start and end dates from the upper level. This layer results in schedules for each single work area. Note that we can treat each single work area separately because of the decoupling effect of the beam-search algorithm.

The lower level is basically formed by work centers. The work centers should implement the schedules obtained from the mid level. In the case of no valid schedules we suggest the use of a modified contract net with proper chosen costs (cf. [6] for details on the algorithm).

The solution of the decision-making problems on the three levels requires communication and, in fact, intra and inter layer coordination across the decision-making units. For example, it is required to postpone information on machine break downs from the manufacturing process up to the upper level in order to update the used models for decision-making.

Agents are an appropriate way for implementing the described production control approach because agents are by definition decision-making entities and have strong communication capabilities that are necessary for implementing coordination schemes.

4 Agentification of the Production Control Problem

Starting with the PROSA architecture [10], we distinguish between decision-making agents and staff agents. Decision-making agents solve decision problems while the staff agents try to support them in the course of the decision-making process. In the PROSA architecture, we find order, product and resource agents as abstract classes.

We identified seven decision-making agent types in our application scenario:

- Each lot agent represents a single lot.
- A PM agent represents a preventive maintenance order.
- Work center agents represent groups of parallel machines on the shop floor. We aggregate work center agents into work area agents. The fab agent consists of all the work area agents.
- Tool agents are used for the representation of auxiliary resources.
- We consider technology agents that encapsulate the product knowledge according to the product holons of PROSA.

Note that for our purposes the difference between holons and agents are not important (cf. [4] for a discussion of related issues).

We define different staff agents that encapsulate the scheduling and monitoring functionality described in section 3. In Table 1, we summarize the basic functionality of the members of the agency that corresponds to the upper and mid level of the hierarchy.

Table 1. Functionality of the Members of the Agency (Fab, Work Area, Work Center Level)

Layer of the Hierarchy	Corresponding Member of the Agency	Task Description
Entire Fab	Fab Agent	- coordination of the work of the fab scheduling agent, the monitoring agent and the work area agents - decision-making in form of sequencing the lots for applying the beam-search algorithm
	Fab Scheduling Agent	- preparation of running the beam-search algorithm - running the algorithm - providing scheduling information
Work Area	Work Area Agent	- coordination of the work of the corresponding work area scheduling and monitoring agent - decision-making in form of choosing the proper machine criticality measure for ABSBH - information providing services
	Work Area Scheduling Agent	- preparation of running the ABSBH - running the heuristic - providing scheduling information
Work Center	Work Center Agent	- implementing the work area schedules in a dispatching manner - mediator in case of the contract-net-type allocation algorithm

5 Architecture of the Multi-agent-System FABMAS

In the next section, we describe the basic architecture of our multi-agent-system FABMAS. During the course of designing the system we used the FIPA Standard and the FIPA Abstract Architecture [2] as an orientation for our own development in order to ensure compatibility to other FIPA compliant multi agent platforms. However, in some cases we deviate from the standard.

5.1 Agent Runtime Environment

According to the FIPA Abstract Architecture for agent systems we developed our own agent runtime environment. The runtime environments consist of an agent di-

rectory service, an agent management system, an agent container and an agent communicator. These parts of the system provide services that are used by the agents to get information about other agents and to communicate and interact with them.

For communication purpose the agent communicator encapsulates communication capabilities. Each communication act between agents is handled by the agent communicator.

The agent directory service is the location where agents register their specification as a service-entry. Agents can ask the local directory service for finding information about other agents they want to interact with. If the information is not available, the directory service tries to find the information by contacting other directory services within the whole multi-agent-system. Hence, it is not necessary to establish a global directory service as a centralized information point in a distributed system.

Each agent runtime environment needs an agent management system that administrates the life cycle of each agent. As a result the management system is responsible for creating the agents, provides potentially mobility services and removes an agent if it is not any longer needed.

The last component of the agent runtime environment is the agent container as a collection of all active agents inside the environment. Figure 1 provides a UML diagram of the runtime environment and its components.

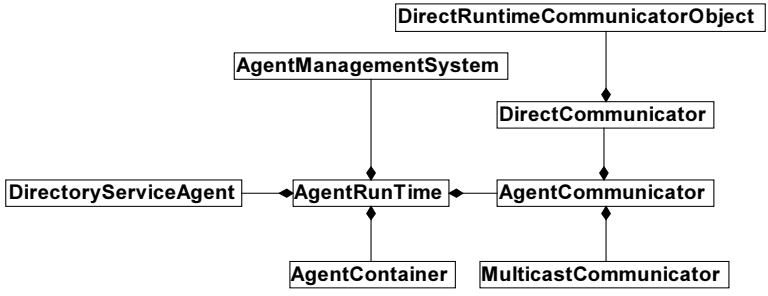


Fig. 1. Agent Runtime Environment of FABMAS

5.2 Generic Architecture for the Agents

Design Approach. From a design point of view we use a role based approach to design different types of agent characteristics. As two basic abstract roles we distinguish between decision-making agents and staff agents as PROSA describes it.

Decision-Making Agents. A decision-making agent is an active part with different decision-making capabilities. We distinguish between resource agents, lot agents, technology agents, and agents for preventive maintenance (cf. Figure 2). The lot

agent is similar to the order holon described in PROSA as well as the technology agent to the product holon. The preventive maintenance agent is specific and provides ongoing maintenance that can be planned.

The resource agent role is the basic role for the fab agent, the work area agent, the work center agent, and the tool agent. The resource agents are the starting point to model the hierarchical control approach described earlier. Each multi-agent-system for production control following our approach has only one fab agent at the highest hierarchy level, one or more work area agents handled by the fab agent at the next level and one or more work center agents within each work area agent.

The hierarchy is modelled by using an agent identifier which is a kind of pointer to the agents. An agent identifier encapsulates information like the agent name, the address where the agent is located and the services provided by the agent. Every agent on a higher level stores the agent identifier of the agents on the next hierarchy level. That means that the fab agent has all agent identifiers of the work area agents to provide them with the necessary information and every work area agent knows the identifiers of the work center agents they are directly responsible for. The way button up works similar, i.e, each work center agent knows his work area agent. Thus, a structure exists that allows for communication and cooperation within the same hierarchy level and between adjacent levels.

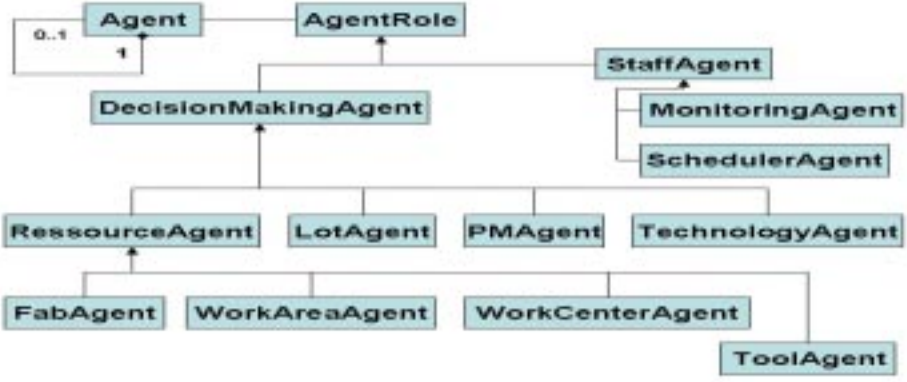


Fig. 2. Agent Hierarchy of the FABMAS Multi-Agent-System

Staff Agents. A staff agent is a special agent that has to support and improve by design the decision-making capabilities of the decision-making agents. Therefore, a staff agent typically encapsulates scheduling and monitoring algorithms.

The architecture of a staff agent should support the following requirements:

- maintaining of a (dynamic) model of the corresponding parts of the hierarchy, i.e., a model of the machines and its capacities and a model of the status of the lots,
- maintaining the storage of at least one (fixed) reference schedule,
- maintaining time issues with respect to scheduling or monitoring, i.e., setting scheduling horizons and time windows for future lot arrivals,

- maintaining and implementing a set of performance measures that fit to the proper model of the corresponding part of the hierarchy,
- maintaining time issues with respect to the (allowed) running time of the solution algorithm,
- maintaining a set of alternative solution methods depending on the problem size and the allowed running time of the algorithm.

5.3 Implementing Communication Abilities of the Agents by Using .NET

By choosing the Microsoft .NET Platform for the development of the agent system various opportunities for implementing communication abilities are given. A similar recent approach using another technology is presented in [9].

An agent communicator is part of every runtime environment and provides two capabilities for communication. The multicast communication is used for announcement of new active runtimes and by the agent directory service to keep their agent list up to date. The direct communication is used for communication purpose between single agents and is implemented by using the .NET remoting framework.

The .NET remoting is a framework for distributed computing. It is ideal for the implementation of communication capabilities of multi-agent systems.

The agent communicator hides all the communication capabilities from the agents and can be used as an interface. An agent that wishes to send a message to an other agent transfers the message to the agent communicator. The communicator determines the location of the agent and if the other agent is on a remote host, the .NET remoting is used for sending the message, if the agent is in the same runtime the message is directly putted into the mailbox of the receiver agent.

The agent communicator is implemented as client and server at the same time and the use of threads makes it concurrently run able. If the agent communicator becomes to be a bottleneck within the communication process because the number of agents rises to a certain level during run time, it is possible to run more than one communicator.

6 Performance Assessment of the Production Control Approach

We suggest a discrete event simulation approach in order to assess the performance of the production control approach. We use simulation models of wafer fabs to mimic the behavior of the shop-floor of wafer fabs. We extend the generic assessment architecture suggested in [8] to the present situation. We use the discrete event simulator AutoSched AP 7.1 that is widely accepted in the semiconductor manufacturing domain. This simulator allows for an easy application of notification/subscription mechanism, i.e., we can react to certain classes of events on the shop-floor. The use of a discrete event simulator as a tool for performance assessment of multi-agent-systems in manufacturing is described in [1]. However, our approach has the advantage that we can replace, in principle, the simulation tool by other information systems on the shop-floor that generates the required information for running the control application, i.e., FABMAS.

6.1 Blackboard Type Data Layer

We adapt the data layer described by Mönch et al. [8] to the present situation. We can differentiate between two types of information in the blackboard, i.e., we have to distinguish between static and dynamic data.

Table 2. Content of the Blackboard

Type of Data	Example
Static Data	- Information about process flows - Setup information - Existing machines - Information on the physical decomposition of the shop-floor - Processing times
Dynamic Data	- Lot release information - Lot states - Machine states - Setup states of the machines

The blackboard is used in different situations.

1. We initialize the blackboard at the beginning of the simulation run by reading all required information from the corresponding simulation objects.
2. The multi-agent-system FABMAS reads information from the blackboard.
3. We update the objects of the blackboard during the simulation run in an event-driven manner.

The blackboard was developed in the C++ programming language. A timer starts the multi-agent-system in the case of time-driven triggering. This approach allows for the application of rolling horizon approaches. On the other hand, events of the shop-floor, i.e., simulation events, trigger the start of the multi-agent-system FABMAS. The multi-agent-system uses only date stored in the blackboard in order to make decisions. No internal information from the simulation model is going to be used. By using this architecture, in principle, the simulator could be replaced by an arbitrary manufacturing execution system (MES) of a real factory. We use the object-oriented database POET in order to make certain parts of the blackboard persistent.

6.2 Communication between Discrete Event Simulator, Blackboard, and MAS

The simulator and the blackboard are implemented in the C++ programming language, whereas FABMAS is implemented using .NET and the C# programming language. The blackboard is encapsulated into a dynamic link library (DLL) that is called by the AutoSched AP simulation engine during the runtime of the simulator.

The communication between the simulator and the blackboard is simple, because the simulator simply calls methods and sends data by using parameters. A direct communication between FABMAS and simulator is impossible and not necessary.

We implemented an interfacing component in order to connect FABMAS with the blackboard. The .NET system supports the encapsulation process and also registers the component into the Windows Registry. The .NET framework creates a special class during run time, the COM callable wrapper. This wrapper offers user specific interfaces to the component and also the typical interfaces of COM. Now, the blackboard can call methods of the interface and transfer data via parameters to the FABMAS multi-agent-system. This data is forwarded by the .NET remoting component to the agent runtime environment. On the other hand, data is transported back from the MAS to the blackboard via method calls using reference parameters. The communication between the described components is shown in Figure 3.

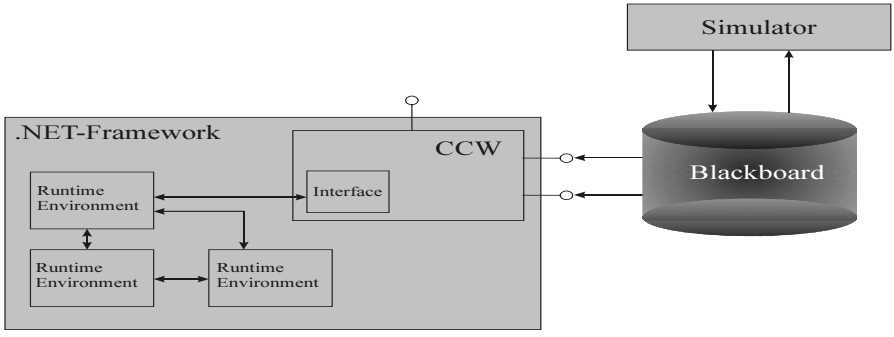


Fig. 3. Overall Architecture of FABMAS and Architecture for Performance Assessment

7 Conclusions

In this paper, we present results on the architecture of an agent-based system for production control of semiconductor wafer fabs. After a brief description of the considered manufacturing process we suggest a hierarchical production control algorithm. We explain why software agents are the proper software artifacts in order to implement the (distributed) production control algorithms. Then we describe the architecture of FABMAS in detail. We demonstrate that it is possible to use the C++ and C# programming language together with the .NET framework in order to implement a hierarchically organized multi-agent-system. We describe how FABMAS interacts with the discrete event simulator AutoSched AP.

In the future, we want to use simulation models from different types of wafer fabs in order to assess the performance of the hierarchical production control approach.

Acknowledgement. This research was supported by a grant from the Deutsche Forschungsgemeinschaft (DFG) Priority Research Program 1083 “Intelligent Agents and Realistic Commercial Application Scenarios”. The authors would like to thank René Drießel and Jie Zou for their valuable programming efforts in course of implementing the FABMAS prototype.

References

1. Brennan, R. W., William, O.: A Simulation Test-Bed to Evaluate Multi-Agent Control of Manufacturing Systems. In: Joines, J. A., Barton, R. R., Kang, K., Fishwick, P. A. (eds.): Proceedings of the 2000 Winter Simulation Conference. Orlando, USA (2000) 1747–1756
2. FIPA Web Site: http://www.fipa.org/
3. Habenicht, I., Mönch, L.: A Finite Capacity Beam-Search Algorithm for Production Scheduling of Semiconductor Wafer Fabrication Facilities. In: Yücesan, E., Chen, C.-H., Snowdown, J. L., Charnes, J. M. (eds.): Proceedings of the 2002 Winter Simulation Conference. San Diego, USA (2002) 1406–1413
4. Mařík, V., Fletcher, M., Pěchouček, M.: Holons & Agents: Recent Development and Mutual Aspects. In: Mařík, V, Štěpánková, O., Krautwurmová, H., Luck, M. (eds.): Proceedings MASA 2001, LNAI 2322. Springer, Berlin Heiderberg New York (2002) 233–267
5. Mönch, L.: Towards an Agent-Based Production Control in the Semiconductor Manufacturing Domain. In: Giambiasi, N., Frydman, C. (eds.): Proceedings of the 13th European Simulation Symposium. Simulation in Industry. Marseille (2001) 331–337
6. Mönch, L., Stehli, M.: Agent-based Modeling and Implementation of Ressource Allocation Scenarios in Manufacturing. In: Urban, C. (ed.): Proceedings Third International Workshop on Agent-Based Simulation. Passau (2002) 149–155
7. Mönch, L., Stehli, M., Schulz, R.: An Agent-Based Architecture for Solving Dynamic Resource Allocation Problems. In: Verbraeck, A., Krug, W. (eds.): Proceedings of the 14th European Simulation Symposium. Simulation in Industry. Dresden (2002) 331–337
8. Mönch, L., Rose, O., Sturm, R.: Simulation Framework for the Performance Assessment of Shop-Floor Control Systems. To appear in SCS Transactions on Simulation. (2003)
9. Ünver, H. Ö., Anlagan, Ö.: Design and Implementation of an Agent-Based Shop Floor Control System Using Windows-DNA. Int. J. Computer Integrated Manufacturing 15 (5) (2002) 427–439
10. Van Brussel, H., J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers in Industry. Special Issue on Intelligent Manufacturing Systems 37(3) (1998) 225–276

A Case Study for Modular Plant Control

Constantin Zamfirescu, Paul Valckenaers, Hadeli Hendrik Van Brussel, and Bart Saint Germain

Katholieke Universiteit Leuven, Mechanical Engineering Department - P.M.A., Celestijnenlaan 300 B, B-3001 Leuven, Belgium
{Constantin.Zamfirescu, Paul.Valckenaers}@mech.kuleuven.ac.be

Abstract. This paper reports on an engineering approach to address the growing need for managing highly modular plants. An implemented agent-based manufacturing controller, inspired by ant social behavior, is presented and discussed. The work aims to provide the key engineering concerns on how the impact of variations in production resources, factory organization and planning processes can be smoothly tackled in respect of plant performance criteria. The insights are drawn along our experience carried out in the MPA (Modular Plant Architecture) project, which is part of the 5th EU framework program.

1 Introduction

The future competitiveness of enterprises depends heavily on their ability to react swiftly to product, process and capacity changes. It will be imperative for them to achieve rapid planning and ramp-up of production structures. This endeavor should be evaluated not merely in aggregated business metrics, but in operational conditions too. To efficiently and constantly take on the manufacturing control aspects, the controller should exhibit self-organizing capabilities in order to face the changes of the production system over its life cycle.

Decentralized control has been already proposed as a feasible way to overcome the limitations that hierarchical and centralized control shows in a highly dynamic environment [1, 2]. Most of the work concerning manufacturing control focuses either on the control algorithms' optimality [3], but using predefined plant configurations, or on the architectural features that enable the required agility in facing the integration aspects [4]. To effectively control a manufacturing plant, both dimensions must be considered orthogonal in evaluating any design approach. Our contribution is not focused on the integration aspects to attain enterprise planning, but on the controller capability to expand over the physical resources as an intrinsic reflection upon the plant modularity. From operative reasons, this capability proved to be an essential requirement to accurately assess any optimization approach.

After a brief introduction of the explored case, the key issues of the implemented agent-based manufacturing controller are subsequently enlightened. The discussion follows the wide-accepted phases in developing an agents-based manufacturing con-

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 268–279, 2003.

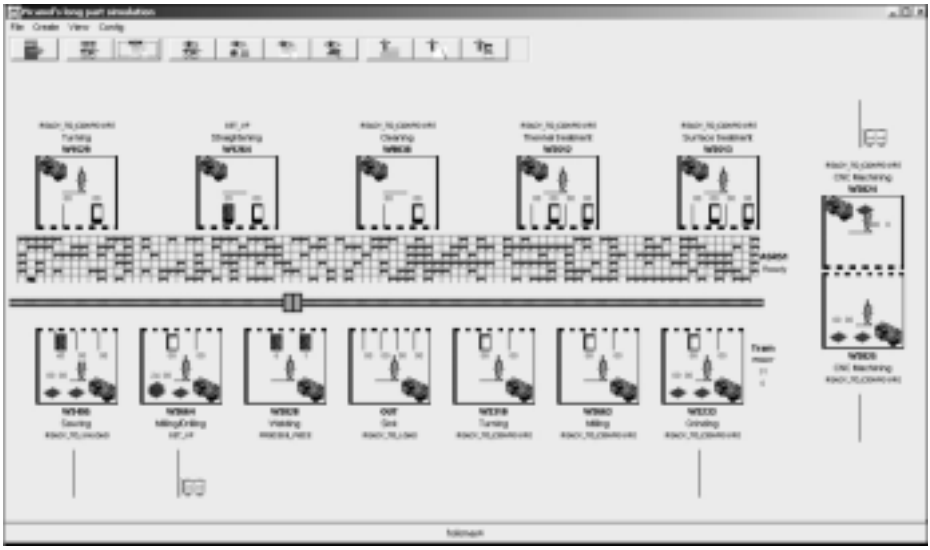


Fig. 1. Screenshot from the PICANOL simulation software

troller [5]: the agent types together with their employed ontologies, the coordination behavior among the agents, and finally the decision-making algorithms. Some concluding remarks and the follow-up work will be given in the last section.

2 The Case Study

The case study concerns the "long parts" department of PICANOL factory, an open job-shop plant for weaving machines components (Fig. 1). The production is organized around an automatic storage and retrieval system (ASRS) that acts as a temporary buffer for pallets shaped as containers. A Tram system is used for storing/retrieving the containers into/from the ASRS and changing the containers at the workstations. Each container contains a variable number of identical parts traveling together till the completion of their processing plan. The machines are grouped in workstations, with a variable number of container docks and with different processing capacity. Typically, a workstation holds two containers: an empty container to be filled with the finished parts, and a full container with parts to be worked on. Inside the workstation a part is taken by the machine operator from the full container and loaded into the processing machines, processed and then unloaded and stored in the originally empty container. When this last container is full, the ASRS is prompted to take it away. Because the Tram has two containers docks, prior to pick up the finished containers it travels to the ASRS to bring the next container that is going to be processed in the requesting workstation. Therefore, once the Tram took the container with the finished parts it unloads the next container without an additional movement. Finished pieces are stored in the ASRS and retrieved in a given number on a daily

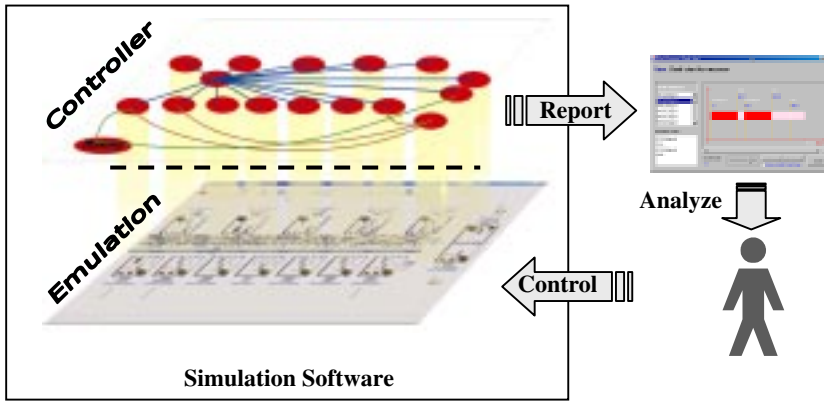


Fig. 2. The human operator as active element in the simulation process

base, according to the assembly orders. There are also some trolleys that can take over the transportation effort as required. The machine operators are assigned to workstations and not to a single machine on the basis of their skills, shifts and preferences. Overall, the plant holds the characteristics of a classical open job-shop, with different alternatives to carry out an operation, either processing or transportation.

3 The General Architecture

Due to the unfeasibility to perform exhaustive and cost-effective investigations on the real plants, most of the experiments in manufacturing control field are studied in a simulated world [6]. To facilitate a smooth integration of the controller (or parts of it) into the real plant, and to preserve the experimental accuracy, the simulation software is divided into two distinct subsystems: the emulation (ES) - mirroring the functionality of the real plant, and the controller (CS) - managing the internal logistics in the production system (Fig. 2). The ES provides to the CS the state of the emulated factory and the control sends the appropriate commands in order to control its state. Being asynchronously connected to the ES, the CS cannot distinguish this from being connected to the real world. The interaction mechanism between the ES and the CS is based on a predefined communication and synchronization protocol (i.e. event notification and command invocation). In this way, the controller can manage any emulation that complies with this protocol. Furthermore, as a replacement of the real plant, the emulation supports, through the user interface, any feasible manipulation that is not restricted by the physical reality (e.g. the movement of a transportation utility, loading the work in-process products into a processing machine, halt of a workstation etc.). User-driven disturbances give the opportunity to get insights on how the system will react to changes that are context dependent and cannot be efficiently captured in the emulation model. Moreover, it gives the users the opportunity to enhance their

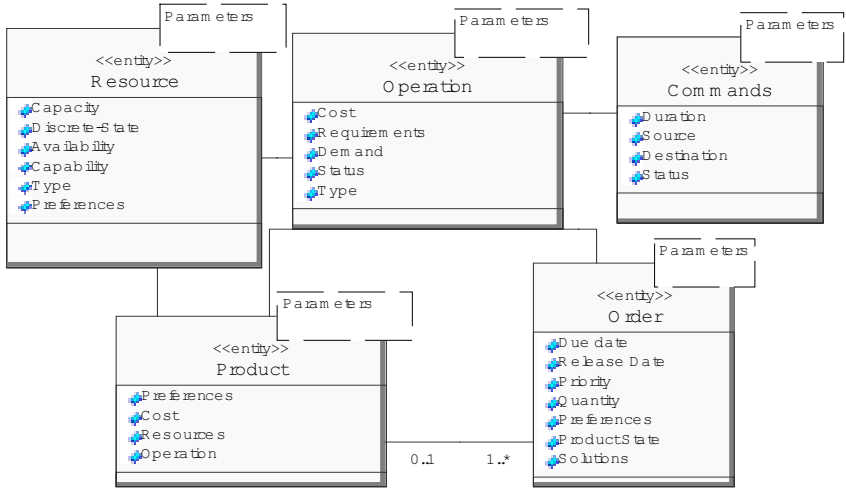


Fig. 3. The basic ontology diagram of the system

knowledge by creating circumstances that are emerging from particular settings and may seldom happen in the real plant. Because this functionally is accessible in real time, the controller should be able to deal not only with disturbances that affect its own performances model (e.g. availability of a resource, introduction of a rush order), but also for the feasibility issues, which usually is considered to be a stable model over the plant topology and configuration.

3.1 Ontologies and Agents

The effort to define a common understanding on the manufacturing planning materializes in a number of attempts [7,8] to extract the key issues that are effectively employed in any intra-logistic optimization problem. In line with most of these general-purpose ontologies, the system design reflects the defined entities at both levels of concern: ES level (i.e. commands or activities) and CS level (i.e. resources, products, operations and orders). The simulation as a whole shares a unified ontology for the static data, while the dynamic behavior of their components is managed disjointedly (Fig. 3) in each subsystem. As part of any manufacturing control, the basic ontology encapsulates knowledge about the inquiring orders, products as subject of orders demand, operations as manufacturing steps to deliver a product, resources that support those operations, and the related activities required to achieve an operation.

The employed PROSA multi-agent reference architecture [2] for manufacturing control reflects closely the entities described in all the ontologies for factory planning. Backed by the ES, it covers all the particularities for controlling the plant at the operational level.

Product. The product entity represents the good or service that can be supplied by the plant. It is embodied in the Product Agent who manages the "product model" of the product type. The product model depends on the product life cycle, preferences and costs constraints. Because in our case, in a given product state different processing plans can be followed, it is represented internally as a tagged directed acyclic graph $P_p = (O, R_p)$, where p stands for each product type of the plant's concern. A vertex from the operations set O, describes an atomic operation owned by the product's recipe. An edge connects an operation to the following one if after its completion the operation outcome corresponds to its assigned tag. In our implementation the Product Agent firstly acts as an information server to the other agents, delivering the right recipes in a particular context. Secondly, it acts as an interface agent for the user, in order to elicit, store and maintain this product related knowledge in a consistent way.

Order. An order is a request for a product that the system can provide. It encloses the customer preferences that drive the system behavior in respect to the factory constraints (i.e. the requested quantity for a deliverable product, release and due date, and priority). Given the customer preferences, an order holds the product state model of the work-in-progress product together with the solution to achieve the desired product type. As an extension to other ontologies, we introduce the order type to distinguish between the customer' orders and intra-logistic orders. Intra-logistic orders denote activities that are driven by the current state of the plant with the aim to change it into a usable/desirable state for the other agents. They include mainly material or facility handling processes, which at their turn need to allocate resources to reach the desired state (i.e. if no empty container or palette is available in the workstation, an intra-logistic order is created by the resource agent attached to that workstation to fulfill these task). The Order Agent reflects in our system the order entity. It is regarded as the work piece with certain control behavior in order to manage its "itinerary" through the factory.

Resource. A resource represents the most abstract class to which an operation could be assigned. It corresponds to a physical part of the manufacturing system (e.g. tram, workstations, ASRS, trolley and machine operators) that exhibits functional capabilities. They are characterized by a predefined operational capacity together with their own constraints and preferences in delivering an operation. Since in manufacturing control the order's decision is strongly affected by the resource type, the ontology was extended to discriminate among processing stations, storage devices and transportation equipment. The resources are represented in the CS by the Resource Agents who take over the responsibility to rule its own allocation procedure. The internal state of a resource is maintained at the ES level but is managed by the Resource Agent by sending the appropriate sequence of commands to the corresponding emulation entity when a certain operation is carried out.

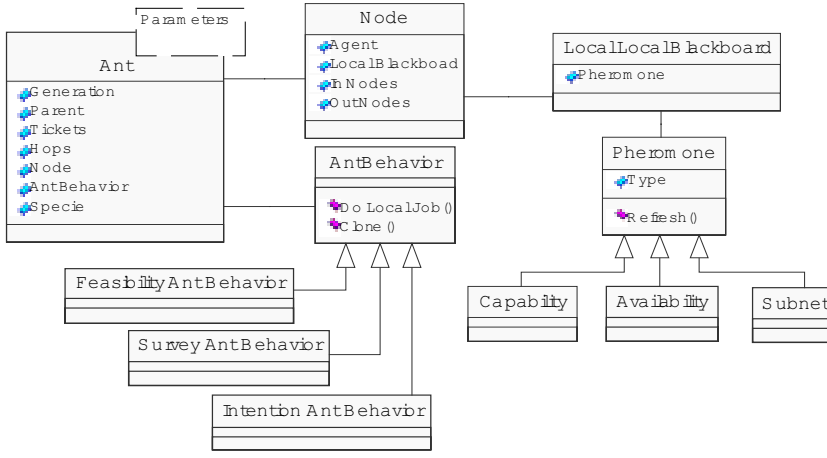


Fig. 4. A sketch of the artificial ants environment data structure

Operation. An operation represents a various set of commands accepted by an emulation entity to realize one operation. Because each command sensibly reflects the duration of the real action (e.g. for the workstation - set-up, load, process, unload, allocate, for the tram – move, load, unload) in simulation time, a probabilistic distribution is employed by the ES to emulate every activity. The cumulative of all these durations represents the probabilistic duration of any operation. The ontology for an operation is also extended to account for the internal operations that transform the product's state as part of the product-processing plan, and external operations that transfer the in-working product from one resource to another. Each operation is decomposed in atomic actions that correspond to the physical activities required to deliver the operation. Only the internal operations belong to the process plan, the others correspond to the intra-logistic process. The system operator as well could send these actions through the user interface.

3.2 Coordination Behavior

In general, coordination between any entities can be done in two ways: coordination by direct communication and indirect coordination within dissipative fields. Because the entities used to describe the ontology of the PICANOL plant reflects real things in the ES, the most appropriate way was to use an indirect coordination method coined by Grassé as stigmergy [9]. It is a form of asynchronous interaction and information exchange among insects mediated by an "active" environment. The coordination mechanism is based on a smelling chemical substance called pheromone. It supports two major operations mediated by the environment in which the insects are situated: aggregation (the cumulative function for the information dropt into the pheromone)

and evaporation (the contraction function for obsolete data). Thus, the pheromone concept equally emphasizes reinforcement and forgetting concerns, as mandatory means to achieve system adaptability.

A significant amount of research results already exist in the area of software inspired by the food foraging behavior of ants. These are mainly ant algorithms for discrete optimization [10]. Most of the work in manufacturing deals with a static environment, where the plant topology and its capabilities are considered to be stable during the optimization process. On the other hand, comparing with the real ants, manufacturing control poses dissimilar constraints. Firstly, there are conflicting objectives in allocating the existing agents. Secondly, in manufacturing control we have to carry out multiple goals simultaneously, which often are not even stable in a given time frame. Thirdly, the individual goal may differ from the society goal; the collaboration terms are agreed upon and not predefined in advance. All of these conduct us to a significant refinement of the underlying principles. Contrasting with the approaches where the ants' behavior is seen as an inspiration source to optimize the system performance criteria, the paper highlight the engineering aspects on how the ants' behavior can inform the engineering practice in building a manufacturing control systems.

Pheromone environment. Generally, ant algorithms operate in a graph environment. For our case the graph denotes the plant topology which is formalized as a tagged directed cyclic graph T = (R, C), where a node R stands for a manufacturing resource and an edge C for a feasible connection between two of them in terms of transferable items (e.g. container, palettes, machine operator) represented as the edge's tag. The CS discovers this topology in the initialization phase by acquiring from the ES the entrances into the plant. Following their connections to the next resources an internal representation of the plant topology is constructed without a prior knowledge upon the plant. For each encountered emulated resource during the plant detection process, the CS creates its counterpart Resource Agent. The agent will generate the local blackboard where the pheromone information will be dropped later. The pheromone environment is constructed around two types of data structure (Fig. 4): resource specific (i.e. resource capabilities, resource schedule) and connection specific (i.e. subnet capabilities). The resource capabilities consist of the operations set which can be performed by the resource. For the machine operator, the structure is extended with their skill level for each operation. The subnet capabilities are attached to every resource's connection and they contains the set of operations which can be processed following that link. These data structures will make possible to have a global overview rooted in the locally available information that will be later used in the agents' decision-making.

Artificial ants. For this case, the ants are implemented as light software agents that are virtually traveling in the navigation graph to accomplish a well-defined task on behalf of its parent agent. The ants are able to sense only the pheromones from the local blackboard attached to the node where they are residing. They also allow for

memorizing the past actions. A number of parameters have been identified as fundamental tuning clues for dealing with loop-detection and preservation of accurate data. These are categorized in (Fig. 3): safety parameters (e.g. the maximum number of nodes an ant can pass and the maximum number of clones an ant can be replicated) and spatial-temporal parameters (e.g. the ant generation and local node of the navigation graph). In each node, the behavior of an ant consists in two subsequent steps. Sensing the locally available information, it firstly tries to interpret/modify it, and secondly to decide where to further clone or not. Because of their simplicity (even the programmer doesn't need to have an overall overview on the problem to implement a new ant), with minimal effort a large space of the desired functionality can be explored in straightforward way [12].

3.3 Decision Making

The decision algorithm take advantage of three ant species created either by the Product Agent who send Feasibility Ants to discover and propagate the feasibility constraints, or by the Order Agent who send Survey Ants to evaluate the processing opportunities as they emerge and Intention Ants to impose the constraints which derives from the short term forecasting of future actions as revealed before by the Survey Ants.

Ants' behavior. The *Feasibility Ants* are created at each possible exit of the plant and for every type of product. They are traveling upstream with an adjustable frequency in order to refresh the feasibility information residing in the subnet capability pheromone. Roughly, this information locally reflects the global ironware constraints (i.e. machines breakdown, availability of the machine operator, introduction of a new product, machine or connection to a new department). Following the feasibility trace, the Order Agent creates *Survey Ants* to discover the best routing alternative to get its product processed. The Survey Ants are traveling downstream from the resource where the Order Agent is currently standing to the next ones in order to evaluate the alternative solution for the remaining recipe. If in the current product state an Order Agent can follow different processing plans, various generations of ants are sent downstream for a selection of the possible recipes. In every node of its journey, the Survey Ant will virtually execute the intended operation to be processed in the Resource Agent. The result of this execution accounts for the current reservation state of the order. Once there are sufficient complete solutions, the Order Agent will send downstream *Intention Ants* to book the resources belonging to the chosen plan amongst the know solutions from the exploration. They are traveling from one resource to another to allocate the resource's schedule. The allocation result is reflected in the availability pheromone, which is updated in conformity. To avoid complexity, the stop condition for any ant resides in the pheromones' updating policy. It accounts for the ant generation (preventing pheromone alteration with

obsolete data) and for the work done by the other ants (if one ant with the same goal already found a better solution there is no need to further explore).

Resource's scheduling. The correct choice of scheduling techniques and rules is highly sensitive to the factory's performance criteria, the particular configuration of the plant as well as to particular knowledge residing in the factory culture. Moreover these rules could vary across different resources and orders. The performance criteria delineated by PICANOL are the increase of throughput, the lead-time minimization and the improvement of labor and resource utilization. The first step was to reflect the heuristic used nowadays in the real plant. These are static rules for the orders as well for the resources. For the orders it is the Earliest Due Date dispatching rule that initially schedule the operations for its process plan and First In First Out for the resources. Due to the occurrence of disturbances, the scheduling is redone on a daily basis to account for the new state in the plant. The agent-based decision algorithm reveals that it is possible to overcome these disturbances in real-time, using the short-term forecasts based on the intentions propagated by the Order Agents and the facts reflected in the pheromone by the Resource Agent. Fig. 5 shows in the upper side how the solution of an Order Agent evolves over time when for example the repairing time for the workstation where it is intending to be processed took less then expected and the priority of the order increased due to its approaching due date. Moreover, because ants are able to account for the future position of the Tram, the transportation time is more precisely predicted in contrast to the current situation where it is totally neglected. The Resource Agent is responsible to rule its dispatching policy against the overall order priority in the system. These rules are applied both when the Survey Ants are virtually executing an operation into the resource and when the Intention Ants are trying to propagate the intention to execute this operation. The second step was to extend the Resource Agent with the facility to use other dispatching rules [9] (either static, e.g. Shortest Setup Time or dynamic, e.g. Minimum Slack). Notice that, because they are the most volatile and plant-specific decisions, the dispatching rules are built in the top of the decision mechanism (no s/w maintenance cascade). Moreover, they are assigned to the real entities in the system with no need to keep homogeneity in employing them (each resource may be governed by its own rules).

4 Benchmarking Concerns

So far the paper described how the social insects behavior could inform the engineering practice in building a flexible manufacturing control system. Their behavior turns the controller into an open to inspection system for optimization purpose. Besides local availability of general control data, the artificial ants emergently build up a short term forecasting of the whole plant based on the agents' intentional state. This chapter will highlight some implications on the benchmarking concerns. In order to understand the consequences of different optimization techniques – either local, at the resource level, or general, at the factory level - the development process follows three

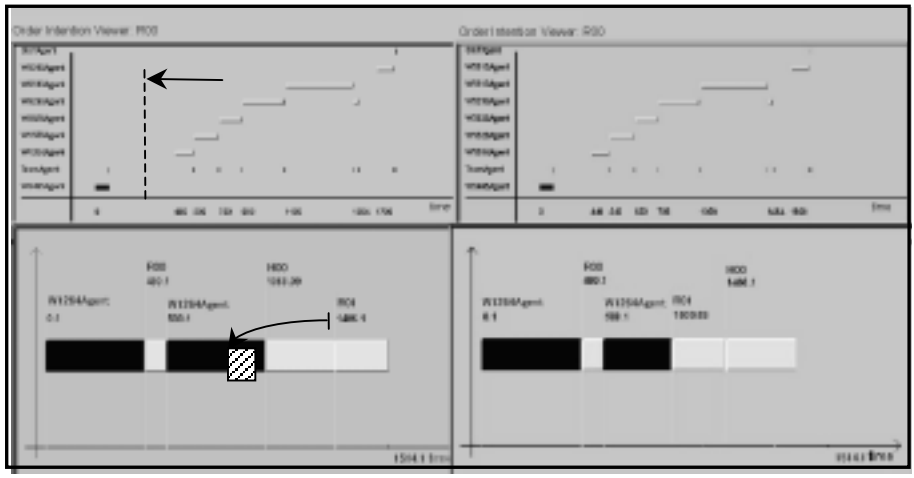


Fig. 5. Solution evolution for an Order Agent and a Resource Agent

distinct stages, namely verification, validation and simulation campaign. They are close related to the work practice (what people actually do) in the real plant, as a holistic reflection of the whole plant.

Verification refers to the process where the implemented model of the plant is performing as it was designed. The main role of this phase is to check if the computer model complies with the assumptions made over its conceptual model. As a result of the propagation of agent's intentions, the designer may get an immediate and overall picture if the conceptual model is comprehensive enough to capture all the required data for controlling propose. All the intra-logistic processes, usually hidden and implicit in a descriptive explanation, are closely disclosed (e.g. supplying an workstation with an empty container, palette's availability for trolleys). In this way the strength and accuracy of the conceptual model is thoroughly tested, with no analysis cascade.

How precisely the model is representing the real plant is carried out during the *validation* phase. The rationale of this stage is to prove if the plan simulation is valid enough to use it in making real decisions. It entails a tight cooperation with the client in order to gain his confidence in your optimization techniques. For the controller this step has two main outcomes. It firstly preserves the tacit optimization knowledge already available in the factory (e.g. the opportunity to share a machine operator among multiple workstations or the use of topological data when storing the containers based on the workstations' proximity) that are at least time-consuming to be discovered during the simulation runs. Secondly, because the experts in the plant optimization find difficult to identify and communicate their knowledge (they are experts in their domain and not in their expertise), the simulation provides a sensible communication means to elicit these knowledge (e.g. why a container will necessarily pass through the ASRS and not directly move to the next workstation). The ant-like engi-

neering approach implicitly brings the dynamic issues of the plant control and not merely the static ones.

After the completion of these steps, an evaluation of any high-level controlling algorithm might be relevant for the targeted case. The *simulation campaign* implies a through investigation of the correlations between different control variables (from the dispatching rules to the nervousness of the system in propagating its processing intentions) and their influence on the plant performance criteria (resource's productivity, orders' lead time, plant's throughput). For the presented engineering approach, this step requires changes at the top level of the system without any danger to undermine its core functionality. It concerns only the system tuning (e.g. criteria to weight the available solutions for a given order, the complexity of exploration space versus awareness and consideration of new alternative solutions) with no cross-alteration of agents' capabilities. The presented case is currently in benchmarking phase, and its evaluation is out of the scope of this paper.

5 Conclusions and Further Work

Continuing our work in holonic manufacturing systems [13], the paper introduced the basic ideas behind the MPA project on how an agent-based manufacturing controller can preserve its functionality against highly customizable modular plants architecture.

Starting from the automatic synthesis of the plant emulation, and continuously uncovering the changes in the plant, the controller is auto-organizing its internal structure (e.g. number of agents, relationship between the agents, communication pattern) in order to cope with these transformations. Following the wide-accepted phases in developing an agents-based manufacturing controller - ontologies, agent types and their behaviour - have been described and explained. The system was fully implemented in Java and proves its ability to cope with disturbances in production resources, factory organization and planning process. From the work practices stance, where the optimality is defined and performance criteria are embedded, the aim of the paper is not strictly focused on the optimisation aspects but merely on how to openly support and maintain a satisfactory performance level. Contrasting with the approaches where the ants' behavior is seen as an inspiration source to optimize the system performance criteria, the paper highlight the engineering aspects on how the ants' behavior can inform the engineering practice in implementing a flexible and robust manufacturing control systems

The presented approach opens several research perspectives that are currently pursued by the authors. The described short-term forecasting mechanism allows system traceability, giving the opportunity to introduce learning algorithms for advanced decision-making. At the resource level, the forecast accuracy can be improved by enhancing the prediction of the execution time of an operation in the specific circumstances. Since the environment is essentially considered part of the solution, the contextual data are automatically captured in the pheromone data structures for further interpretation. Together with the heuristics employed for the resource scheduling, they are the most sensitive and plant-specific decisions that are built in the top of the

system. In this way preserving the system reusability during its life cycle, more sophisticated and better-performing prototypes are subsequently deployed. At the system level, a series of tuning parameters, like the frequency of ant propagation, evaporation rate and level of commitment in assigning a resource, are at this time manipulated from the user interface. We are currently considering an auto-tuning mechanism that will enable an as optimal as possible setting for these parameters in order to attain the global equilibrium in the specified context.

Acknowledgments. This work was supported by the European Commission under the GROWTH program and the K.U.Leuven research council (GOA–$AgCo_2$)

References

1. Parunak, V.D.: Manufacturing Experience with the Contract Net. In: Huhns, M.N. (ed.): Distributed Artificial Intelligence. Pitman (1987) 285–310
2. Shen, W., Norrie, D.H.: Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. Knowledge and Information Systems 1 (1999) 129–156
3. Dorigo, M., Di Caro, G.: The Ant Colony Optimization Meta-Heuristic, New Ideas in Optimization. McGraw- Hill, London (1999) 11–32
4. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P: Reference architecture for holonic manufacturing systems: PROSA. Computers In Industry 37 (1998), 255–274
5. Van Dyke Parunak, H.: "Go to the Ant": Engineering principles from natural multi-agent systems. Annals of Operation Research, Special issue on Artificial Intelligence and Management Science 75 (1997) 69–101
6. Molina, A., Al-Ashaab, A.H., Ellis, T.I.A., Young, R.I.M., Bell, R.: A Review of Computer-Aided Simultaneous Engineering Systems. Research in Engineering Design, 7 (1995) 38–63
7. Smith, S.F., Becker, M.A.: An Ontology for Constructing Scheduling Systems. Proceedings of AAAI, Spring Symposium on Ontological Engineering (1997)
8. Saucer, J.: Knowledge-Based Systems techniques and Applications in Scheduling. Knowledg-Based Scheduling Techniques in Industry. In: Jain, L. (ed.): Intelligent Techniques in Industry, CRS Press (1999)
9. Grassé, P.P. : La theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs. Insectes Sociaux 6 (1959)
10. Bonabeau E., Dorigo, M., Theraulaz, T.: From Natural to Artificial Swarm Intelligence. Oxford University Press, New York (1999)
11. Steegmans, E., Holvoet, T., Janssens, N., Michiels, S., Berbers, Y., Verbaeten, P., Valckenaers, P., Van Brussel, H.: Ant Algorithms in a Graph Environment: a Meta-scheme for Coordination and Control. In: Hanza, M. (ed.): Artificial Intelligence and Applications. ACTA Press (2002) 435–440
12. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Prentice-Hall, Englewood Cliffs, New Jersey (1995)
13. Valckenaers, P., Van Brussel, H., Kollingbaum, M, Bochmann, O.: Stigmergy in Holonic Systems. In: Proceedings of the 5th International Conference on Autonomous Agents, Workshop on Holons: Autonomous and Cooperative Agents for Industry (2001) 15–21.

Implementation of Mobile-Agent-Based Network Management Systems for National Broadband Experimental Networks in Taiwan*

Li-Der Chou, Kun-Chang Shen, Ko-Chung Tang, and Chi-Chia Kao

Department of Computer Science and Information Engineering,
National Central University
Chungli, Taoyuan, Taiwan 32054 , R.O.C.
cld@csie.ncu.edu.tw

Abstract. The paper presents a web-based network management system, using the mobile agents technology, designed for Taiwan's National Broadband Experimental Networks (NBEN). The mobile agent technology is applied to gather and monitor the network status, and is capable of offering efficient, flexible and intelligent network management functions. Through the friendly graphical interfaces, the network status and related information can be easily collected and displayed dynamically. In the future, more mobile agents with intelligence will be designed and implemented, and the system will be transferred to commercial networks.

1 Introduction

Traditional network management systems almost adopt request/response model, such as Simple Network Management Protocol (SNMP), to acquire the network-related information. Such centralized model may generate a large amount of traffic, and the network may be congested by the management traffic. The larger is the size of the network, the more serious is the phenomenon. That is, traditional request/response-based network management systems are not efficient enough at the usage of network bandwidth. Besides, modern networks are much more complex than before, for the interconnection among various types of networks and the diverse characteristics of the multimedia traffic. Thus the modern network management system must be more powerful, flexible and efficient than before. And how to design and implement such network management system is getting important [1].

Mobile agents have been successfully applied to network management 2 3, for mobile agents have the features of autonomy, social ability, reactivity, pro-activity, mobility, security, and veracity 4. Many mobile agent systems have been developed, such as Voyager 5, Concordia 6, Grasshopper 7 and Aglet 8 9. Mobile agents can be

* The research was supported in part by MOE Program of Excellence Research under grant A-91-H-FA07-1-4, and by National Center for High-Performance Computing, National Science Council of the Republic of China under grant NSC 90-2219-E-008-005.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 280–289, 2003.

dispatched to specific network nodes and accomplish the assigned management works as requested. The mobile agents are able to proceed with the assigned management works, regardless of the impact of link failures 10. Besides, the network bandwidth wasted in the network management can be further reduced, compared to the traditional client/server- based network management systems 11. Moreover, the distributed architecture provides an efficient way to detect network faults 12, save the network bandwidth and update the management policies.

The paper adopts the mobile agent technology to design and implement the web-based network management system (WNMS) for National Broadband Experimental Networks (NBEN) 13in Taiwan. Through the mobile agents, several network nodes can share network management works for load balancing, and thus the response time is reduced 14. Mobile agents can be dispatched to network nodes and to gather the network status. In the WNMS system, all network information can be acquired, by clicking appropriate position of the network topology shown in the friendly graphical interface.

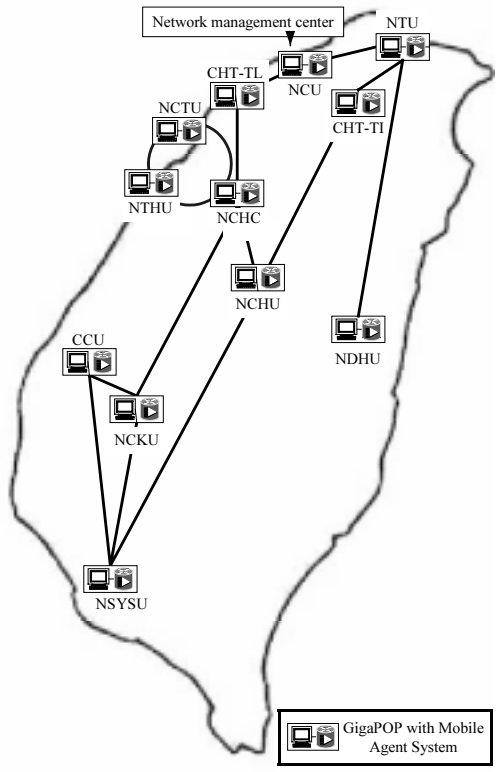


Fig. 1. Network topology of Taiwan's National Broadband Experimental Networks.

2 System Design

The WNMS system is designed and developed for Taiwan's NBEN network. The NBEN network was established by National Center of High-Performance Computing, National Science Council, Taiwan, R.O.C. in 1999, and consists of 12 Points of Presence (POPs): nine main national universities, National Center of High-Performance Computing, Chung-Hwa Telecommunication Laboratory, and Chung-Hwa Training Institute. The objective of the NBEN network is to provide a flexible and QoS guaranteed broadband network environment, and provide a testbed for experiments on newly derived multimedia network applications and advanced communications protocols. Figure 1 shows the 12 POPs in NBEN, and each POP consists of a Nortel BayNetworks 5000 BH router and a Fore Runner ASX-1000 ATM switch. Because it is hard to embed the agent execution environment (AEE) in both kinds of the routers and the switches without the assistance of the venders, we connect a personal computer, the trusted host or the so-called mobile agent system (MAS), to each POP, as shown in Fig. 2. The MAS provides AEE to execute mobile agents delivered from other POPs or issued by the attached POP. The architecture of the MAS is shown in Fig. 3. The interaction between the MAS and the POP is based

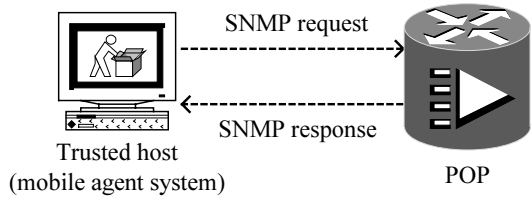


Fig. 2. The trusted host associated with a POP provides the agent execution environment.

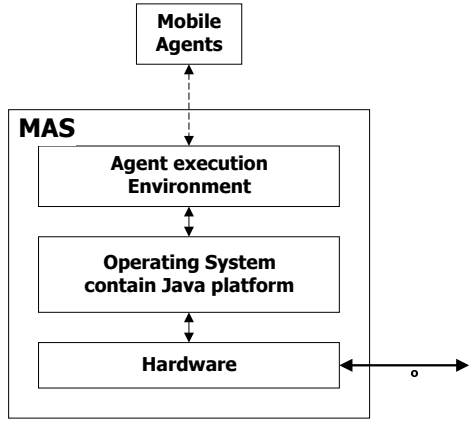


Fig. 3. Architecture of the MAS system

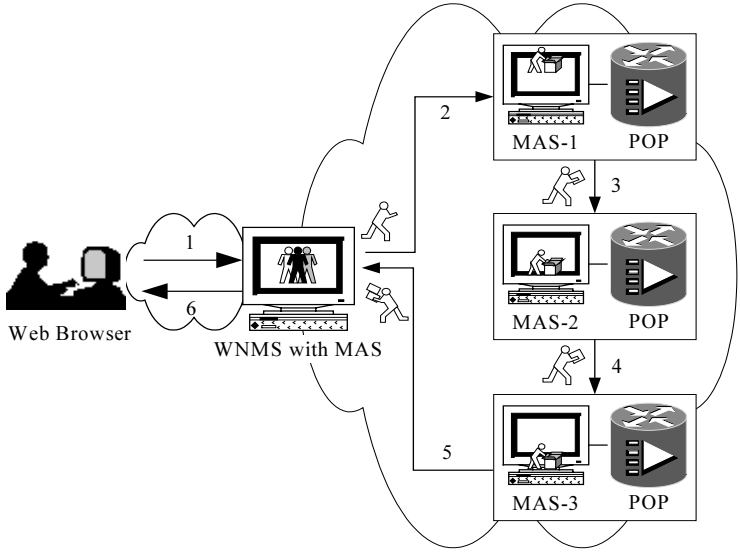


Fig. 4. System structure and operation procedure of the WNMS system

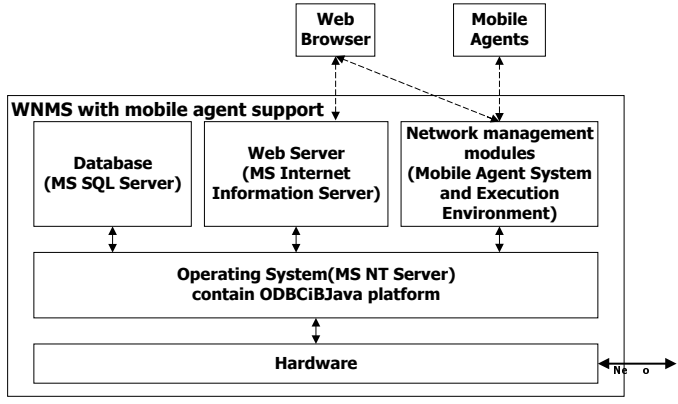


Fig. 5. Modules in the WNMS server with MAS support

on SNMP, for information gathering, and telnet, for parameters adjustment. For the reason of security, the MAS is physically connecting to the corresponding POP. In addition, if the MAS cannot physically connect to the POP, authentication and encryption mechanisms must be provided between them, so that the MAS can be trusted. The WNMS system is designed to be able to

(1) show the network topology on the graph user interface

(2) show the current status of ATM switches by clicking appropriate icons on GUI.

(3) authenticate each request
(4) issue, dispatch and execute mobile agents
(5) provide communication mechanisms and communication protocols between the system and mobile agents.
(6) query the values of the specific object identifiers in management information base
(7) generate reports for the requests

Figure 4 shows the system structure and the operation procedure of the WNMS system. A WNMS server is established so that network managers can access the web-based network management system through web browsers. Besides, for network management works are implemented by mobile agents, the WNMS server is responsible for issues, dispatches and execution of mobile agents. After network managers log remotely into the WNMS server, and a specific network management function is requested, the WNMS server will dispatch the corresponding mobile agent to the MASs of the POPs sequentially and perform the required network management functions. The results will be carried to the WNMS server by the mobile agent, and then displayed on the web browser. The collected data are also stored in the database. Figure 5 shows the modules in the WNMS server: database, web server, and network management module. The web server offers the monitoring interface, and the downloading interface for the applet of the user interface. The database stores the basic system and network status information. The network management module provides all network management functions that are executed on the AEE of the mobile agent system. The mobile agent system is able to dispatch mobile agents and provide the communication interface to mobile agents. For the communication between the network management modules and the mobile agents must rely on the mobile agent system, the network management module has to be running on the mobile agent system. All of the above modules are implemented on Java platform.

Figure 6 shows the signaling procedure. Customers' web browsers have to support Java and can download the applets of the graphical user interfaces from the web server. The web server and the network management server at the system side provide all system functions and services. Applets are executed at the customer side to take the responsibility of the interaction between users and the server. The network management server is responsible for retrieving the network information and dispatching mobile agents.

There are five submodules in the network management module of the WNMS system: (1) the network information gathering submodule, (2) the identity authorization submodule, (3) the add/delete POP node submodule, (4) the PVC application submodule, and (5) the create/delete PVC/PVP submodule. The network information gathering submodule is capable of collecting the system and network information by issuing, dispatching and managing mobile agents. The identity authorization submodule is responsible for user identity authentication and the authorization management. The add/delete POP node submodule allows the network managers to modify the database that stores the NBEN network topology by adding or deleting POP nodes on the graphical user interface. The PVC application submodule provides an online application form for users to apply permanent virtual circuits

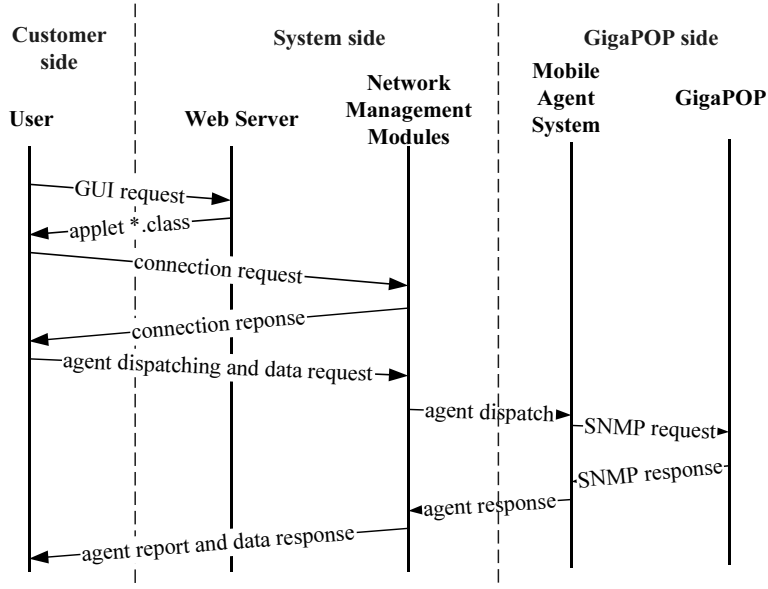


Fig. 6. Signaling procedure of the WNMS system.

(PVC) in ATM networks and a scheduler to check whether the residential network resources are enough for the application or not. The create/delete PVC/PVP submodule is responsible for the creation or deletion of PVC or permanent virtual paths (PVP). For modules (3) (4) and (5) are related to the control of network resources, authentication and authorization of submodule (2) are required before accessing the three submodules. According to the provided functions of each submodule, only submodules (1) and (5) are necessary to adopt the technology of mobile agents. Thus the two submodules will be illustrated in the section.

3 System Implementation

The network management system server is implemented on a computer with Intel Pentium III 450MHz CPU, 128 MB memory, and 10GB hard disk space. The operating system is Microsoft Windows 98. The web server adopts Microsoft Internet Information Server 3.0 and the database system adopts Microsoft SQL Server 7.0. The system is developed primarily by Java, and uses C language to write the ICMP programs to probe the network. IBM Aglets 1.1 beta 2 is the tool for the development of mobile agents, and it also provide the mobile agent system named Tahiti. The programming language is Java for the security mechanism of Java virtual machine is more secure.

Figure 7 shows the network environment for the WNMS system implemented in the NBEN POP node at National Central University (NCU), where the mobile

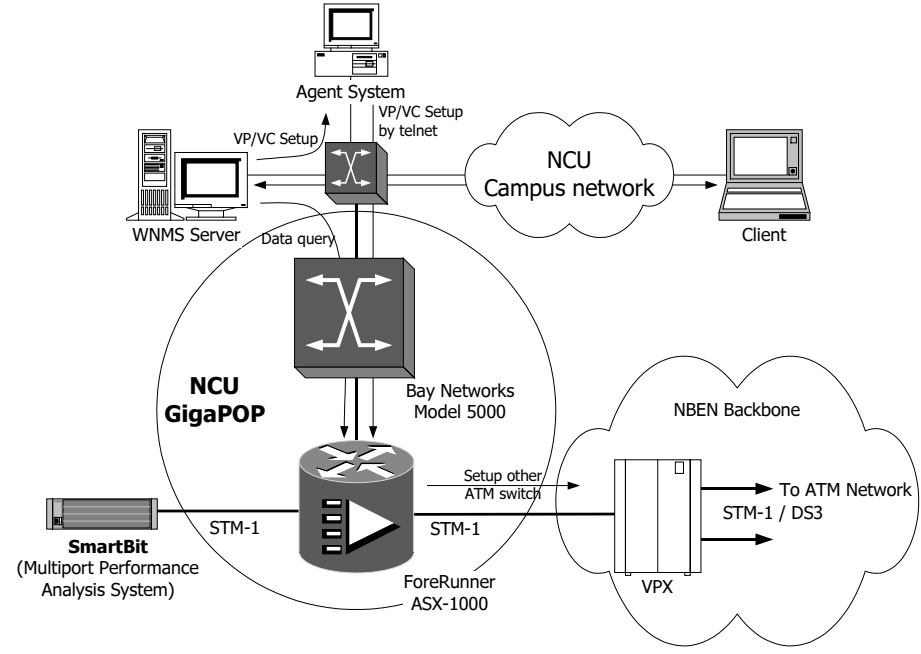


Fig. 7. Network environment of the WNMS system implemented in the NBEN POP at National Central University

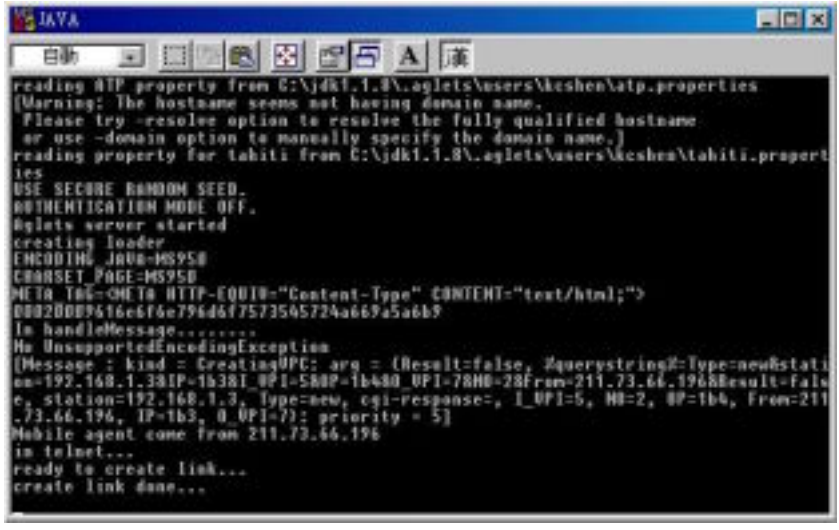


Fig. 8. Execution of a mobile agent on the trust host for the creation of a VPC

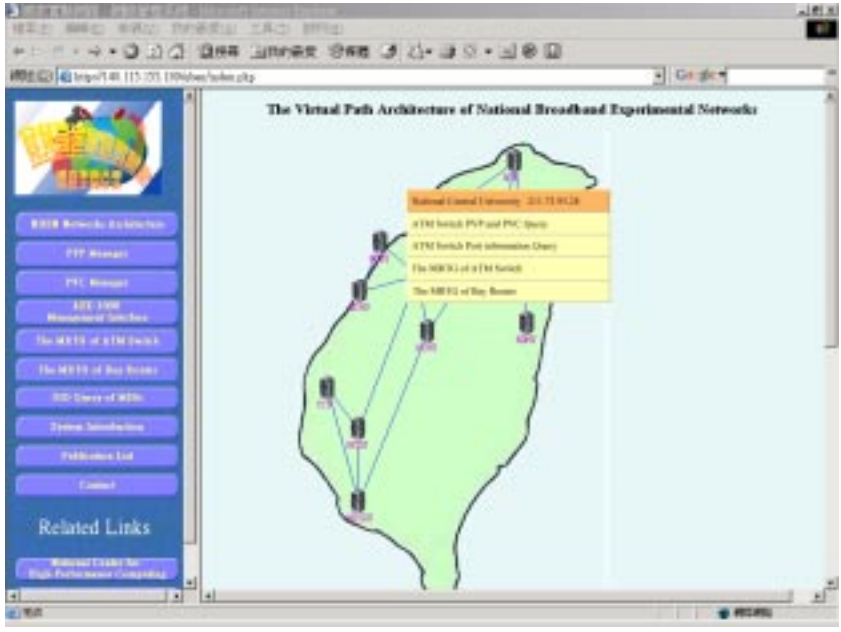


Fig. 9. Homepage of the WNMS system

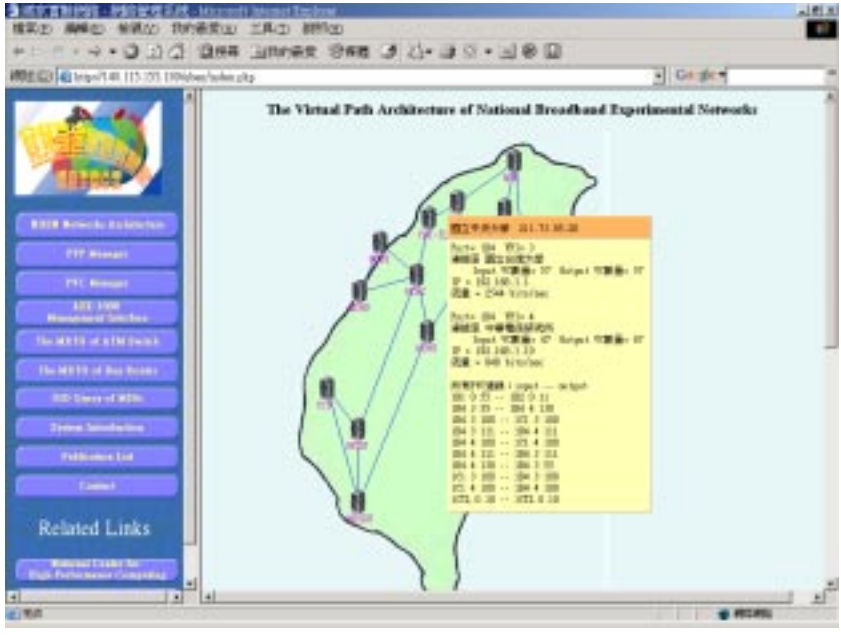


Fig. 10. Results for the PVP/PVC query of ATM switches

agent system module shown in Fig. 5 is installed in an individual computer. Also the SmartBit multiport performance analysis system is adopted to verify the success of the creation and bandwidth setup, using mobile agents, for virtual paths or virtual circuits. The screen for the execution of a mobile agent on the mobile agent system to create a VPC is shown in Fig. 8.

Figure 9 shows the homepage of the implemented WNMS system for the NBEN network, where the VP topology is depicted. As the cursor is moved on a POP in the management system interface, the management functions associated with the POP will be given on the screen. Four functions are provided: ATM switch PVP/PVC query, ATM switch port information query, MRTG of ATM switches, and MRTG of BayNetworks routers. As the ATM switch PVP/PVC query is clicked, the information related to all PVPs and PVCs passing through the POP will be listed, as shown in Fig. 10. Besides, the amount of the instant traffic for each VP is also given in the figure.

4 Conclusions

In the paper the web-based network management system using the mobile agent technology for Taiwan's NBEN network is designed and implemented. The mobile agent technique is applied for the features of intelligence, efficiency, flexibility and activity. Mobile agents are issued and dispatched to POPs in NBEN to monitor dynamically the network status and setup PVP/PVC. Experimental results show that, using the mobile agent technology, the required time for routine monitoring functions may increase, not reduce. However, the mobile agents for on-demand services, such as setup PVC/PVC, simplify the management works and win the applause of network managers. In the future, more intelligent mobile agents will be developed to assist the management of the NBEN network, and the system will be applied to the management of commercial networks.

References

1. A. Puliafito and O. Tomarchio. "Advanced network management functionalities through the use of mobile software agents," Proceedings of IATA '99 – the 3rd International Workshop on Intelligent Agents for Telecommunication Applications, Stockholm, Sweden, Aug. 1999.
2. W. J. Buchanan, M. Naylor and A. V. Scoot, "Enhancing network management using mobile agents," Proceedings of the Seventh IEEE ECBS 2000 International Conference and Workshop on the Engineering of Computer Based Systems, Edinburgh, U.K, pp.218–226, Apr. 2000.
3. S. Tao, Y. Cao, J. Yin and N. Xu, "A mobile agent based approach for network management," Proceedings of WCC–ICCT 2000 – International Conference on Communication Technology, Beijing, China, pp. 547–554, Aug. 2000.
4. V. A. Pham and A. Karmouch, "Mobile software agents: An overview," IEEE Communications Magazine, vol. 36 pp. 26–37, Jul. 1998.
5. ObjectSpace – Voyager 4.0, http://www.objectspace.com/products/voyager/.
6. Concordia – Java Mobile Agent Technology, http://www.concordiaagents.com/.

7. Grasshopper – the Agent Development Platform, http://www.grasshopper.de/.
8. IBM Aglets Software Development Kit, http://www.trl.ibm.com/aglets/.
9. A resource for Java Aglet community, http://www.aglets.org.
10. D. B. Lange and M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison Wesley, Aug. 1998.
11. D. Zhang, "Network management using mobile agent," Proceedings of ICCT'98 International Conference on Communication Technology, Beijing, China, pp. S43-11-1-5, Oct. 1998
12. P. Zhang and Y. Sun, "A new approach based on mobile agents to network fault detection," Proceedings of 2001 International Conference on Computer Networks and Mobile Computing, 2001, Los Alamitos, CA, USA, pp. 229–234, Oct. 2001.
13. National Broadband Experimental Network, http://www.nben.net.tw.
14. Marcelo G. Rubinstein, Otto Carlos M. B. Duarte and Guy Pujolle, "Using mobile agent strategies for reducing the response time in network management," Proceedings of WCC–ICCT - 2000 International Conference on Communication Technology, Beijing, China, pp. 278–281, Aug. 2000.

An Agent-Based Simulator for Electricity Markets: Seller, Buyer, and Trader Players

Isabel Praça[1], Carlos Ramos[1], Zita Vale[1], and Manuel Cordeiro[2]

[1] GECAD – Knowledge Engineering and Decision Support Research Group
Institute of Engineering / Polytechnic Institute of Porto
R. Dr. António Bernardino de Almeida, 431
4200-072 Porto, Portugal
{isp, csr}@dei.isep.ipp.pt, zav@dee.isep.ipp.pt
[2] Secção de Engenharias
Universidade de Trás-os-Montes e Alto Douro
Vila Real, Portugal
cordeiro@utad.pt

Abstract. As electricity markets evolve, there is a need for new modelling approaches that simulate how electric power markets could evolve over time and how participants in these markets may act and react to the changing economic, financial and regulatory environment in which they operate. To study electricity markets behaviour and evolution we propose a multi-agent simulator where agents represent several entities that can be found in electricity markets, such as generators, consumers, market operators and network operators, but also entities that are emerging with the advent of liberalization, such as traders. The simulator probes the possible effects of market rules and conditions by simulating the strategic behaviour of participants. In this paper a special attention is devoted to the behaviour and characteristics of Seller, Buyer and Trader agents, highlighting their strategies and decision processes in order to gain advantage facing the new emerging competitive market.

1 Introduction

Many electricity markets are undergoing a transition from centrally regulated systems to decentralized markets. The unbundling of the generation, transmission and distribution functions that is part of this evolution creates opportunities for many new players or agents to enter the market, such as power brokers, traders and load aggregators. As electric utility systems around the world continue to move towards open, competitive markets, the need for new modelling techniques will become more obvious.

The main advantage of Multi-Agent Simulation is to allow the modelling of decision processes and actions of individual agents (e.g. consumers, generators, regulators), rather than the aggregate system behaviour patterns and trends, as in traditional approaches. To gain insights into decentralized electricity markets, we developed a Multi-Agent Simulator. Unlike traditional tools, our Agent-based Simulator does not

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 290–301, 2003.

postulate a single decision maker with a single objective for the entire system. Rather, agents, representing the different independent entities in Electricity Markets, are allowed to establish their own objectives and decision rules. Moreover, as the simulation progresses, agents can adapt their strategies, based on the success or failure of previous efforts.

Some agent-based simulators have been constructed for Electricity Markets: PowerWeb [1], where demand is always fixed and just single uniform auctions are studied; the Auction Agents for the Electric Power Industry [2], which only implements a Dutch Auction; the Simulator for Electric Power Industry Agents (SEPIA) [3], which only implements a bilateral contracts market. Very relevant is the work of John Bower [4], Monclar [5], and Nicolaisen [6] however they are interested in studying only a particular market, the England and Wales market. These models have hinted at the potential of agent-based models for the analysis of Electricity Markets. Another important application of Simulation is presented in [7], where the authors focus specially on the hydroelectric power stations parameters. In [8] an interesting lab experience shows the practical utility of Electricity Market Simulators.

Our simulator has some different and relevant characteristics: it is intended as a Decision Support Tool, so it includes several types of negotiation mechanisms usually found in Electricity Markets, such as Bilateral Contracts, Symmetric and Asymmetric Pools, and Mixed Markets, to let the user test them and obtain sensibility about the best way to negotiate in each one. It includes agents representing several entities: ongoing entities and emerging ones, such as traders. The different agents have their own objectives and strategic behaviour. Particularly important is the strategic behaviour of agents representing suppliers, consumers and traders. To obtain an efficient decision support, agents have the capability of using an algorithm that analyses different bids under several scenarios (we call a pair bid-scenario a *play*). These ideas originate the development of a prototype of the proposed Multi-Agent Simulator [9].

This paper starts by a description of the general features of the Multi-Agent Simulator and then a special attention is devoted to some of the agents: Seller, Buyer and Trader agents, highlighting their strategies and decision processes.

2 Overview of the Multi-agent Simulator

The market simulator we propose is related to the electricity spot market, with 24 negotiation periods each day. The simulator is flexible, the user defines completely the model he wants to simulate: how many agents are in the model, the type and strategies of each one and the type of market. Three types of markets are simulated: Bilateral Contracts, Pool Market and Mixed Market, where both types of negotiations are possible. There are different types of agents in our model: Market Facilitator, Sellers, Buyers, Traders, Market Operator and Network Operator Agent. In this section we will describe their roles, functionalities and the interactions between them. A special highlight will be given to Seller, Buyer and Trader agents in sections 3 and 4.

The Market Facilitator plays the role of market coordinator of the Electricity Market and its main goal is the correct functioning of the market. It knows the identities

of all the agents present in the market, regulates the negotiation process and assures the market is functioning according to the established rules. Agents before entering the market must first carry out the registration with the Market Facilitator, specifying their market role and services. This approach is also a first step to support a future architecture for an electronic marketplace where agents will negotiate with each other substituting the real entities involved in electricity markets.

Seller and Buyer agents are the two key players in the market, so a special attention is devoted to them, and particularly to their objectives and strategies they can use to reach them. Seller agents represent entities able to sell electricity in the market, e.g. companies holding electricity production units. Buyer agents represent electricity customers and electricity distribution companies. The number of Seller and Buyer Agents in each scenario is completely defined by the user, who must also specify their intrinsic and strategic characteristics. By intrinsic characteristics we mean the individual knowledge related to reservation and preferred prices, and also to the available capacity (or power needs if it is a Buyer agent). By strategic characteristics we mean the type of strategies the agent will employ to reach the objective of selling the available capacity at the best price, if the agent is a Seller, or to buy the needed power if the agent is a Buyer. Seller Agents will compete with each other, since they are interested in selling all their available capacity and in obtaining the highest possible market quote. On the other hand Seller Agents will cooperate with Buyer Agents while trying to establish some agreement that is profitable for both. This is a rich domain where it is possible to develop and test several algorithms and negotiation mechanisms for both cooperation and competition.

The increase in competitiveness creates opportunities for many new players or agents to enter the market; one of these players is the trader. The introduction of this new entity, with well-defined responsibilities, allows liberalization and competition in the electricity industry to be developed and simplifies the way the whole process works with producers and customers on the market and the relationship with the market operator. This entity participates in the market on behalf of customers. It is an intermediary between them, who delegate on the trader the purchasing of their needs, and the suppliers. Its behaviour, and interactions with other agents, is described in detail in section 4. One important feature of our simulator is the inclusion of this type of agent, usually not considered in related works.

Bilateral contracts are agreements between a single Seller and a single demand agent, either a Buyer or a Trader. The establishment of this type of contracts is made through a series of requests for proposals that are initiated by demand agents. If a demand agent chooses to participate in the bilateral market it will start by sending a request for electricity, exposing its price expectations. This request triggers the negotiation process and is delivered to all Sellers in the simulated market. In response, a Seller Agent will analyse its own capabilities, current availability, past experience, and checks its technical feasibility, through the Network Operator Agent feedback. Then, it formulates a proposal, if it is able to make an offer to the requested parameters, and sends a message to the source agent, specifying its proposal. Demand agents evaluate the received proposals and either accept or reject the offers. Details about message handling can be found in [10]. On the basis of the results obtained in a ne-

gotiation period Seller, Buyer and Trader agents revise their strategies for the next period.

The Market Operator is responsible for the power exchange (Pool) mechanism. This agent is only present in simulations of Pool or Mixed Markets. It will send a request for proposals, organize the received bids, determine the market price and select the accepted and rejected bids. After the processing of all bids, and market price established, the results are communicated to each pool participant. Bids matching process is done with the technical approval of the Network Operator Agent.

In Pool markets the most common type of negotiation is a standard uniform auction [11][12]. If only suppliers are able to compete in the Pool, it is called an Asymmetric Market. If both suppliers and buyers are able to compete, elastic demand, it is called a Symmetric Market, also known as Double Auction in the auction theory. Both of these types of pool mechanisms are included in our simulator. In mixed markets agents must decide whether to establish a bilateral contract before trying the pool, or just after Pool results if bids were not accepted. To make this type of decision agents use their past experiences and market strategies (see details in section 3.1 and 3.2).

The Network Operator Agent represents the transmission grid and all the involved technical constraints. One Network Operator Agent is present in every simulation and every contract established, either through bilateral contracts or through the pool must be communicated to it, which analyse its technical viability. In Pool Markets the Market Operator is the agent responsible for dealing with the Network Operator Agent, in order to obtain technical approval for the bids matching process. In a bilateral contract, the Seller Agent must be sure of the technical possibility of delivering energy to the Buyer Agent localisation, so it is responsible for talking with the Network Operator agent before an agreement is reached.

3 Seller and Buyer Agents

The development of a strategic offer that ensures high profitability is a fundamental issue for the market participants, so the policies implemented by each agent must be analysed carefully. Both Seller and Buyer Agents have strategies to define the price at which they are willing to sell (buy). To obtain an efficient decision support, these agents have the capability of using an algorithm that analyses different bids under several scenarios, we call a pair bid-scenario a *play*. The analysis of several *plays* permits the construction of a matrix with possible results, and then, after a decision method is applied, the agent can select the bid to propose, having an idea of the expected payoff. This will increase its performance and improve their negotiation capabilities.

The strategies and the algorithm for agents Decision Support will be detailed in this section. These agents have similar structure and a kind of symmetrical (due to their antagonistic objectives) behaviour, for this reason they are both treated in this section, however, whenever necessary the differences between them will be pointed out.

3.1 Strategies

Strategies to change the price under a negotiation period, also referred as time-dependent strategies, are called: *Determined*, *Anxious*, *Moderate* and *Gluttonous*. The difference between these strategies is the time instant at which the agent starts to modify the price and the amount it changes. *Determined* agents will maintain their prices constant during the negotiation period. *Anxious* agents will start modifying the prices early in the negotiation period but by a small amount. *Moderate* agents will start changing the prices in the middle of the period by a small amount, and *Gluttonous* agents will only start changing the prices at the end of the negotiation period but by a major amount.

To adjust price between negotiation periods, also referred as behaviour-dependent strategies, two different strategies were implemented: one called *Composed Goal Directed* and another called *Adapted Derivative Following*. The *Composed Goal Directed* strategy is based on two consecutive objectives, the first one is selling (or buying) all the available capacity (power needed) and then increase the profit (reduce the payoff). Seller agents that use this strategy will decrease price if in the previous period the available capacity, the first objective, was not completely sold and will increase price when all the available capacity was sold in the previous period, trying to maintain satisfied the first objective while trying to obtain a higher profit. For Buyer agents the strategy is symmetrical: a Buyer will increase price if in the previous period the power needed was not completely bought and will decrease price when all the power needed was obtained, trying to maintain satisfied the first objective while trying to decrease market price in order to spent less money.

The *Adapted Derivative Following* strategy is based on a *Derivative Following* strategy proposed by Greenwald [13]. The *Adapted Derivative-Following* strategy adjusts its price by looking to the amount of revenue earned in the previous period as a result of the previous period's price change. If last period's price change produced more revenue per good than previous period, then the strategy makes a similar change in price. If the previous change produced less revenue per good, then the strategy makes a different price change.

The price adjustment is based on the same calculation for both strategies and takes into account the difference between the desired results and the obtained results in the previous period. The calculation will be exemplified for a Seller, however for Buyer agents the calculations are identical but with different parameters, regarding Buyers objective of buying all the needed power at the lower expensive price. The price for the next period (1) will be the previous period price adjusted by some amount (2), that will increase or decrease the previous price according to the strategy used.

$$price_{i+1} = price_i \pm amount_{i+1} \tag{1}$$

$$amount_{i+1} = price_i * \left(\beta + \frac{\Delta_i}{capacity\_available_i * \alpha} \right) \tag{2}$$

$$\Delta_i = capacity\_available_i - energy\_sold_i \quad (3)$$

Instead of adjusting the price each day by a fixed percentage (like β), the change is scaled by a ratio based on the objective of selling all the available capacity (3). The amount price changes increases with the difference between the power intended to sell and the power actually sold. β and α are just scaling factors.

These strategies were implemented and some results were already obtained. To explain the differences between the two behaviour-dependent strategies we will describe a short example.

Example. This example concerns a small scenario with few Buyers and Sellers that is simple but adequate to illustrate some conclusions about the behaviour of the implemented behaviour-dependent strategies. The scenario has one Seller more competitive than the others and two Sellers with very similar prices. So, it is possible to analyse the behaviour of a kind of monopoly situation, in periods of less demand, where the only Seller able to sell is the most competitive one, and a competition situation, in periods of higher demand. 24 periods were simulated, so it was possible to analyse the effect of demand fluctuations during a day.

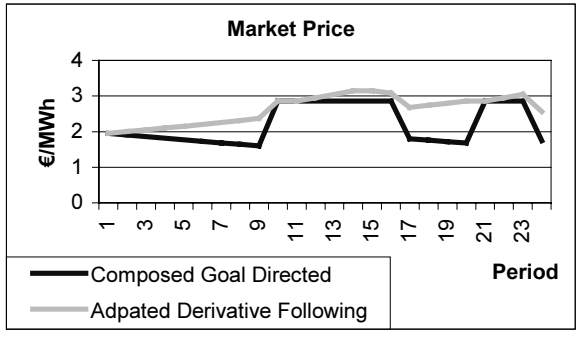


Fig. 1. Asymmetric Pool.

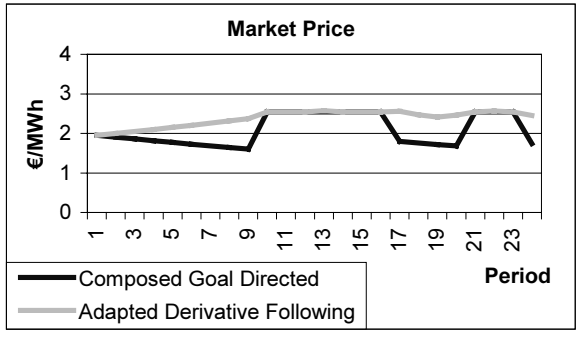


Fig. 2. Symmetric Pool with non-strategic Buyers.

About the behaviour of the two strategies, as we can see in Figures 1 and 2, the *Adapted Derivative Following* strategy obtained higher market prices than the *Composed Goal Directed* strategy. After a careful analysis of the results we can conclude that when demand is less than the total available capacity of the most competitive Seller, it will be losing money when using the *Composed Goal Directed* strategy, since with this strategy it will be decreasing price, trying to sell more, which is not possible because demand is not enough. However, the *Composed Goal Directed* strategy can be important, particularly to obtain more market share when two or more Sellers compete directly, because of their similar proposed prices, since with a slight price modification one Seller agent can have the chance to sell more. It seems interesting to develop another strategy that combines the two described. The idea of combining the two strategies is to give an agent the possibility of selecting the most suitable strategy for each period, based on market conditions. For example, if a Seller concludes that demand is lower than its available capacity, it will use the *Adapted Derivative Following* strategy. On the other hand, if it concludes that there is a competitor with similar prices then it will use the *Composed Goal Directed* strategy.

3.2 Decision Support Algorithm

This algorithm is particularly suitable for markets based on a Pool or for Mixed markets, to support Seller and Buyer agents' decisions for proposing bids to the Pool and accepting or not a bilateral agreement.

To assure an efficient Decision Support, Seller and Buyer agents can analyse different probable scenarios and evaluate the expected returns. Sellers and Buyers are like players in a game, and this algorithm will analyse the possible scenarios resultant from other agents' reactions. Then, a decision evaluation method will be applied in order to decide what to propose in a Pool, and how to negotiate in mixed markets, i.e., what to bid in the Pool and whether a bilateral contract should be accepted or not. Each agent has historical information about other agents in the market and demand forecasts. Agents build a profile of other agents containing the probable proposed prices, limit prices and capacities. With this information, several scenarios may be analysed to obtain conclusions about the best way to deal with competitors and how to bid to obtain a good, or the most reliable, payoff. The definition of the proposed bids and scenarios to be analysed is important.

For each market player there will be two prices {*limit_price*, *probable_price*}, where *limit_price* will be the minimum price, if the player is a Seller, and maximum price, if a Buyer, and *probable_price* will be the previewed proposed price. The number of scenarios, which result from the different arrangements that it is possible to establish considering the two prices for each agent, is given by formula (4), where n is the number of other agents in the model.

$$n^2 \tag{4}$$

On the other hand it is necessary to define which bids, or which move, should an agent, or player, analyse. The agent should analyse the incomes it is capable of ob-

taining by bidding its limit and desired prices and by bidding prices, higher (or lesser, if it is a Buyer) than its limit, that are nearby other agents' proposals, but are a lit bit smaller (or higher, if a Buyer). So, the agent will try to compete with others by proposing prices that are closer to their proposed and limit prices, but are sufficiently smaller to overcome them.

The algorithm will be detailed for an agent that represents a Seller agent. The analysis for a Buyer agent is symmetrical, since the limit price is a maximum price, and not a minimum, and the objective is to buy at the slightest price, while a Seller objective is to sell at the highest possible price.

Let *j* be the agent that is doing the analysis, cap_j its available capacity, $limit\_price_j$ its minimum acceptable price and $desired\_price_j$ its desired price. Let P denote the set of all the players, Sellers and Buyers, in the market. Lets say ε is the smallest positive number allowed as a bidding increment. The bids agent *j* must analyse are:

$$\begin{cases} \text{bid}(\text{limit\_price}_j, \text{cap}_j) \\ \text{bid}(\text{desired\_price}_j, \text{cap}_j) \\ \text{bid}(\text{limit\_price}_i - \varepsilon, \text{cap}_j) \\ \text{bid}(\text{probable\_price}_i - \varepsilon, \text{cap}_j) \end{cases}, \forall\ i \in P, i \neq j \tag{5}$$

subject to:

$$limit\_price_i - \varepsilon > limit\_price_j \text{ AND } probable\_price_i - \varepsilon > limit\_price_j$$

The maximum number of bids to analyse happens when $limit\_price_j$ is smaller than every other agent probable or limit price and is given by 2∗n+1. We will call a *play* to a pair bid–scenario, so, the total number of plays to analyse is *bids_number*scenarios_number*, and the maximum value it can achieve is:

$$(2 * n + 1) * n^2 \tag{6}$$

In a model with few players the number of *plays* can be high, for example with 6 players, to do this analysis a maximum of 275 *plays* needs to be analysed! But, is it important to analyse every possible scenario or just the most disadvantageous and the most probable ones? Since our Multi-Agent Simulator is intended as a Decision Support tool, the user should have the flexibility to decide which scenarios, and how many, are important to analyse. To do so, the user must define the scenarios to be simulated by specifying the price that other agents will propose. That price is given by:

$$Price_i = \lambda * Probable\_Price_i + \varphi * Limit\_Price_i \tag{7}$$

where λ and φ are scaling factors that can be different for each agent.

Suppose that the user selects λ=0 and φ=1 for every Seller and λ=1 and φ=0 for every Buyer, this means she/he is interested in analysing a pessimistic scenario. But, if she/he selects λ=1 and φ=0 for every Seller and Buyer, she/he is interested in ana-

lysing the most probable scenario! With this formula the user can define for each agent the proposed prices for every scenario she/he wants to be considered.

A table, similar to the one presented, will be constructed with the results obtained for each *play*.

Table 1. Results of the analysed *plays*.

	Scenario 1	Scenario 2
bid 1	23	45
bid 2	14	50

Suppose Table 1 represents the results obtained for two possible bids through the analysis of two scenarios, where Scenario 1 can be, for example, a pessimistic one, and Scenario 2 a probable one. The decision method that a Seller agent uses to decide which bid to propose is the MaxiMin method (Buyer agents use MiniMax method). With this method the selected bid will be the one with the highest minimum value, in the table given the chosen bid will be *bid1*. This method gives a "pessimistic" view; however, it is the most safe, particularly when there are some uncertainties. This analysis gives the agent not only decision support about the best bid to propose in a Pool but also makes possible the improvement of the negotiation mechanism for establishing bilateral contracts, because based on it counter-proposals between Seller and Buyer agents can be included.

4 Trader Agents

Traders are emerging in the context of liberalization as intermediaries between consumers and suppliers. Consumers have incentives to become members of an alliance guided by a Trader. One of them is the fact that consumers can gain market power by grouping their purchases; another one is that some skills and market knowledge are required to be an efficient player in the market, and not all consumers will have the ability or interest in dominating those issues, while Traders will necessarily be specialized in the field. Particularly, domestic consumers will be, at least in an initial phase of market liberalization, interested in being represented by Traders, since they will not be experts in dealing with the new market rules and behaviour, and probably will not have enough time or interest to become specialized in the Market.

These entities are included in our model as Trader Agents, who establish Bilateral Contracts, by dialoguing with Sellers, and/or participate in the Pool, representing several Buyer Agents. The inclusion of this entity is an important research topic in studying the evolution of electricity markets. Figure 3 illustrates message exchanged between Trader agents and the other agents in the model. The protocol for establishing Trader set of clients, at the right part of the figure, will be detailed in this section, the protocol for establishing Bilateral Contracts and for trading in the Pool is similar to the defined for Buyers and detailed in [10].

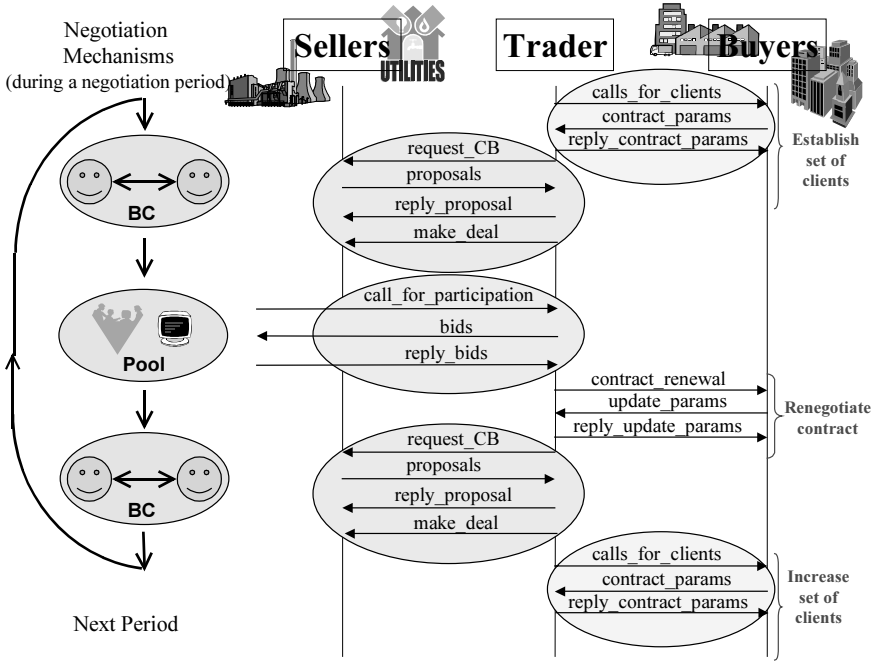


Fig. 3. Trader agents message exchange.

The simple negotiation protocol for establishing a Trader set of clients, involves the Trader (T) and a set of Buyers, possible customers of the Trader, $B=\{b_1, b_2, \ldots b_n\}$.

1. First T advertises B to represent them on the market. T → B "calls_for_clients" message.
2. Each $b_i \in B$ considers whether to make a contract with the T and defines its parameters. $b_i \rightarrow$ T "contract_params" message.
3. T evaluates parameters and replies to every b_i who answered its request. Usually T accepts every client, unless its parameters are unrealistic. T → B "reply_contract_params" message.

Steps 2 and 3 may have several iterations, until the agents reach an agreement or one of them gives up.

When the time limit of a contract is approaching, the Trader contacts its clients in order to negotiate its renewal. If a client, from the set of clients $C=\{c_1, c_2, \ldots c_k\}$, is interested in the renewal it will send a message to the Trader specifying the parameters of the contract renewal, such as the contract duration and other updates such as new consumption needs.

1. First T advertises $c_j \in C$ to renew the contract that is finishing. T → c_j "contract_renewal" message.
2. Then, $c_j \in C$, if interested in renewing the contract, will contact T to update the previous established parameters. $c_j \rightarrow$ T "update_params" message.

3. T replies to c_j acknowledging the contract prolongation with the updated parameters. T → c_j "reply_update_params" message.

Steps 2 and 3 may have several iterations, until the agents reach an agreement or one of them gives up. Modifications on Traders set of clients, such as ceased contracts or renewals and increase on the number of clients, usually take place during a negotiation period, however these modifications are only put into practice on the subsequent negotiation period.

Traders negotiate on the market, either through Bilateral Contracts or in the Pool using the same message exchange as defined between Buyers - Sellers and Buyers - Market Operator, described in Figure 3 and detailed in [10]. However, Traders evaluation function and decision analysis process is different and takes into account the contract parameters established with its clients to fulfil them while trying to obtain some profit. An important distinction is that Traders are forced to assure the total demand of their clients is satisfied, even if for that they loose money. Traders also have strategies and a decision support algorithm similar to the defined in section 3 for Sellers and Buyers.

5 Conclusions

We propose an agent-based simulator that can be a valuable framework to study new electricity market rules and behaviours and to analyse their possible evolution.

The agent-based simulator permits combinations of bilateral trading and power pool markets. In fact, the possibility of simulating several types of markets, and not just a particular one, is an important feature of our simulator. Another important aspect is the strategic behaviour of Seller and Buyer agents. It is not common to find applications that consider Buyers with strategic behaviour and that can be a drawback in obtaining warrantable results, particularly in markets with Symmetric Pools. The strategies for bid definition are another contribution of our work, although other strategies including learning and adaptive behaviours of agents are currently under study. The agents' strategies were analysed and detailed, particularly the Decision Support Algorithm based on Game Theory for analysis and bid selection. This algorithm seems to be very promising. The inclusion in the model of entities that are emerging in decentralized electricity markets, such as Traders, is another important feature of our simulator, since it permits to gain insights into the evolution of the market and behaviour of these new entities.

This simulator is also a first step to support a future architecture for an electronic marketplace where agents will negotiate with each other substituting the real entities involved in this kind of electricity markets. The electric power industry provides a very rich domain for illustration, but there are many other areas where these ideas could also be fruitfully applied.

References

1. PowerWeb – URL: stealth.ee.cornell.edu/powerweb/
2. Electric Power Research Institute – URL: www.agentbuilder.com/Documentation/EPRI/
3. Harp, S., Brignone, S., Wollenberg, B.: SEPIA: Simulator for Electric Power Industry Agents. IEEE Control Systems, vol 20, nº 4 (2000) 53–69
4. Bower, J., Bunn, D.: Agent-based Simulation – An Application to the New Electricity Trading Arrangements of England and Wales. IEEE Transactions on Evolutionary Computation, vol.5, nº5 (2001) 493–503
5. Monclar, F. R., Quatrain, R.: Simulation of Electricity Markets: A Multi-Agent Approach. Proc. of the 2001 International Conference on Intelligent Systems Application to Power Systems (2001) 207–212
6. Nicolaisen, J., Petrov, V., Tesfatsion, L.: Market Power and Efficiency in a Computational Electricity Market with Discriminatory Double-Auction Pricing. IEEE Transactions on Evolutionary Computation, vol.5, nº5 (2001) 504–523
7. Villar, J., Rudnick, F.: Hydrothermal Market Simulator Using Game Theory: Assessment of Market Power. IEEE Transactions on Power Systems, vol.18, nº1 (2003) 91–98
8. Contreras, J., Conejo, A. J., Torre, S., Munoz, M. G.: Power Engineering Lab: Electricity Market Simulator. IEEE Transactions on Power Systems, vol.17, nº2 (2002) 223–228
9. Praça, I., Ramos, C., Vale, Z., Cordeiro, M.: A Multi-Agent Simulation Prototype for Decision Support in Electricity Markets. Proc. of the 2002 AI, Simulation and Planning in High Autonomy Systems (2002) 247–252
10. Praça, I., Ramos, C., Vale, Z., Cordeiro, M.: Intelligent Agents for Negotiation and Game-based Decision Support in Electricity Markets. To appear in Proc. of the 12th Intelligent Systems Applications to Power Systems Conference (2003)
11. Sheblé, G.: Computational Auction Mechanisms for Restructured Power Industry Operation. Kluwer Academic Publishers (1999)
12. Shahidehpour, M., Yamin, H., Li, Zuyi: Market Operations in Electric Power Systems: Forecasting, Scheduling and Risk Management. John Wiley & Sons (2002)
13. Greenwald, A., Kephart, J., Tesauro, G. :Strategic Pricebots Dynamics. Proc. of the First ACM Conference on Electronic Commerce (1999) 58–67

AgenTec – Concepts for Agent Technology in Automation

Arnulf Braatz, Michael Höpf, and Arno Ritter

Fraunhofer Institute Manufacturing and Automation (IPA),
Nobelstraße 12, D-70569 Stuttgart, Germany
ritter@ipa.fraunhofer.de

Abstract. This paper presents approaches and solutions for agent-based manufacturing systems developed within the *AgenTec* project. The developed technologies, methods and tools for automation systems will realise *"plug and produce"*.

1 Introduction

Automation systems are characterized by a high degree of heterogeneity and the requirement for easy engineering as well as robust operation. Heterogeneity is defined by different types of machines (such as assembly robots, mobile manipulators, autonomous mobile robots, robot assistants, AGVs, conveyor belt systems etc.), by different control systems (e.g. PLC - IEC 61131-3 or robot control systems), and by different communication systems (e.g. field bus or ethernet).
In order to manage this heterogeneity and the arising challenges of new markets, technologies and processes, several agent-based manufacturing paradigms have been developed in the past ten years. Examples for these paradigms are *Holonic Manufacturing Systems*, *Random Manufacturing Systems* and *Bionic* or *Genetic Manufacturing Systems* [5,6,14]. Even approaches like *Fractal Factory* or *Mass Customization* can be seen in the context of agent-based systems. Much more integral approaches like *Transformable Enterprise Structures* [15] can also be performed by agent technology.
The latest achievements of agent technology for manufacturing systems have reached some respectable degree: First agent-based industrial applications have already been installed and run successfully, e.g. an agent-based control system of Schneider Automation for motor engine assembling at DaimlerChrysler [6]. Furthermore, agent based robotic systems have been developed in the academic sector, e.g. within the *AgenTec, DIAMOND* and *PABADIS* project [1,2,7]. An experiment for the combination of different Multi Agent Systems (MAS) consisting of mobile and industrial robots was realized by Schneider Automation and Fraunhofer IPA in Stuttgart, Germany. It showed that the integration effort for implementing control systems was lower compared to traditional approaches. These results were presented on Hanover Fair 2001 [10].

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 302–309, 2003.

The main advantages for manufacturing systems provided by agent technology are expected by *"plug and produce"*. This means easier engineering (planning, development, implementation, ramp up, customization, maintenance, de-construction etc.) and more robust operation with high flexibility in terms of changes such as disturbances, integration of new systems and subsystems. Indeed, these intended achievements are correct and challenging, but the reality of today's development processes and life cycle of agent-based manufacturing systems is somehow different from theses goals and demands. The reasons are as widespread as the specific requirements for agents [12]: There is no well-suited integral approach for developing agent-based automation systems.
Our intention is to identify these gaps and to offer some solutions for the development process of technical agents. In the next chapter we will give an overview of development process for technical agents. In chapter 3 and 4, we consider the main development steps – agentification, detailed design and implementation including a material flow scenario - ending with a short summary in chapter 5.

2 Process for Developing Technical Agents

Most approaches in this area are focused on the operation of agents or on development processes. This can cover a wide range of activities starting from layout planning of factories, over material flow planning and construction to design of control systems. The know-how of engineering and computer sciences can be applied on this problem. Taking these aspects into account, an approach for the development process considers the phases showed in Fig. 1.

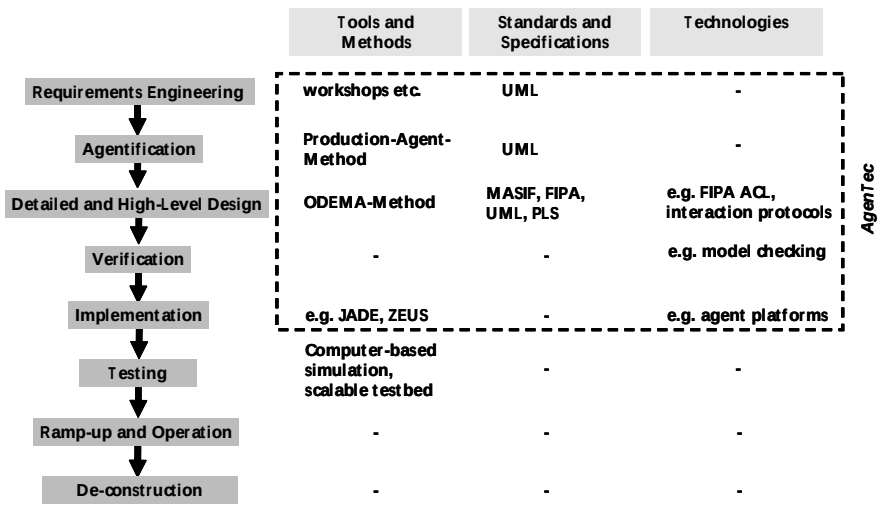


Fig. 1. Development process for agent-based automation systems

The development process is based on development methods, tools, relevant standards and specification as well as technologies such as agent platforms including agent

management services, interaction protocols, agent languages etc. [3,4,5,8,9,16]. In addition, each step of the development process offers its own specialized methods, standards and technologies (Fig. 1).
We assume that the development of agent-based control for manufacturing systems is very similar to software-engineering, because the same technologies and methods are partially used. Thus the development process starts with finding the functional and non-functional requirements. This could be done by sharing workshops with system-users and customers. In this context, the agentification is a special step of developing technical agents. Based on an agent model with certain properties, parts of an existing or planed manufacturing systems are grouped as the physical part of an agent (e.g. Production Agent Method [10,11]). With the help of this top-level agent model, functional requirements and a simple description of the manufacturing process can be transformed into an integrated model (e.g. Unified Modeling Language *UML*, [16]).
Furthermore, the design phase is based on agent models in terms of decomposition by these top-level agent models down to control-units (e.g. Function Blocks). Typically, the design phase is iterative and has different levels of details [12]. In the case of using a formal agent model a verification based on mathematical principles can be applied after designing.
Finally, the technical agents, existing so far as software models, have to be realized with those hardware parts found during agentification. Because agents have in principle a non deterministic behavior, testing strategies and methods (e.g. test cases, simulation) known from software engineering are not sufficient.
Because of these requirements, some new approaches are proposed. These are:

- Agentification (identification and modelling) of subsystems and components as agents, e.g. industrial robots or mobile robots as technical agents;
- Modelling of processes, e.g. assembling as MAS interactions;
- Choice of a suited agent specification (agent language, agent platform) or software modelling paradigm;
- Agent models;
- Technical concepts for an agent platform and the integral MAS;

As figure 1 shows, we have focussed on the phases agentification to implementation. Some points are still open and need to be researched in the future. These are integral approaches for the verification of distributed, decentralized systems, the ramp-up phase, the maintenance phase or the de-construction phase of agents.

3 First Step: Agentification Based on *Production Agent (PA)*

The *agentification* is the process of identification and modelling of systems working in a plant e.g. robot systems, machining center, conveyer belts etc. [11]. The objective of such a process or method is it to find logically and physically independent sub-systems and components (*Agent Body*). That means their mechanical parts and low-level control-software (logic- and motion control, user interfaces etc.) have to be found and grouped as an agent body that can be controlled by an agent-based control, so called *Agent Head*. It is assumed that such a method of agentification leads to an agent-based software architecture, which enables the

implementation of a robust and flexible control system (High-Level Design). Generally there are several possibilities to cluster autonomous and cooperative systems and components as agents. Mobile (autonomous and cooperative) robots such as *KURT* or *Care-O-Bot* are able to be very easily defined as agents considering their characteristics [13]. From this point of view also mechanically not coupled systems (milling stations, machining center etc.) can be treated as agents. Because this approach is not systematic, a new concept for a agentification had to be created within the *AgenTec* project.

The solution is a method based on the physical and logical dependencies including the aspects of co-operation and autonomy (so called p- and a-aggregation). In this context, an agent is an autonomous entity with the ability for cooperative decision making, e.g. negotiation. P-aggregation means for example: A simple sensor or actuator is a physical part of a mechatronic system and thus a p-aggregation connect it to another component of this system. In this context a-aggregation means that a component or subsystem cannot execute its manufacturing function without another. For example a machine cannot perform tap-drilling without drilling a hole before [10,11]. In principle the method handles only parts which are persistent regarding the manufacturing process and are able to execute a manufacturing function.

In terms of how to use this method for applying, there are two different ways. The first option is a *Top-down* approach and the second one follows the *Bottom-up* approach.

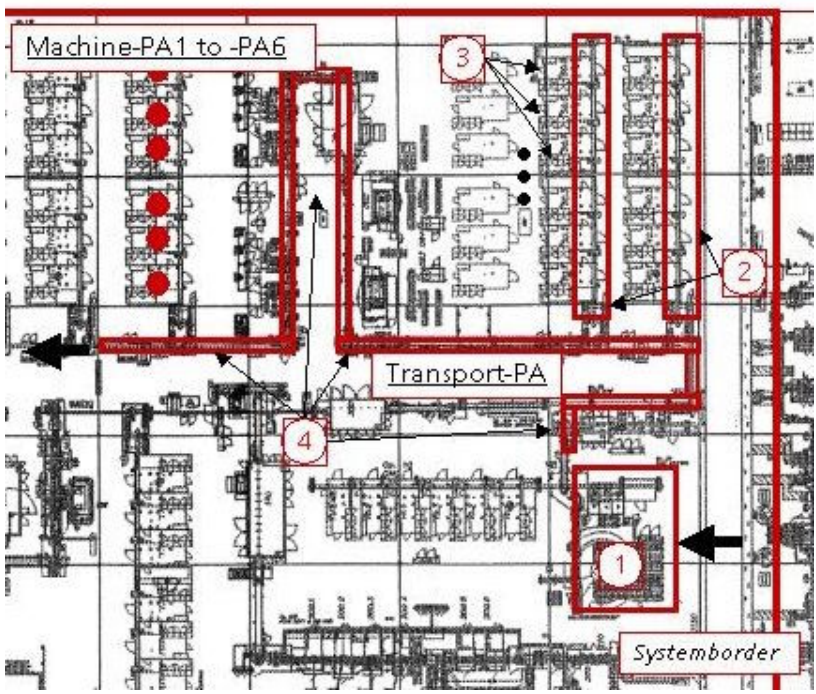


Fig. 2. Agentification of a plant (layout: motor engine assembly)

An idea of how the Top-Down option works is shown in Fig. 2 as part of a factory layout. Objective of this plant is it to mill cylinder heads for motor engines. For this purpose this manufacturing system consists of a material flow system (conveyer-belts (4), palletising robot (1), and gantries (2)) and rows of milling tools (six per row (3)). The thick arrows shows the direction of work pieces. The conveyer-belts have to shift these work pieces through the whole plant, that means from gantry to gantry. Each milling tool is loaded and unloaded by a gantry. The agentification method starts with modelling the top-stage agent which is the manufacturing system itself. In a second step the top-stage agent is divided in a couple of agents which still fulfil the requirements of a PA. Steps of dividing PA can go on further until elementary, not dividable agents are found. Additionally with the help of UML, communication links and message protocols can be modelled in this context, which finally leads to an agent-based control architecture [16].
Applying this method to the plant example yields to kinds of PA. The first one is the *machine agent* that consists of one single milling station (for example machine-PA1 to PA6 marked by the bold dots in Fig.2). The second agent type is built of all systems that are responsible for the work piece flow. Because of the fixed line-structure further steps of decomposition are not possible.
Moreover within the *AgenTec* project a new taxonomy for agents was designed in order to manage the role of human beings within manufacturing systems [9]. The human itself is playing different roles within the production, e.g. as „server“ or „client“. He is the one who is making decisions, developing and working on the production. His role differs from the roles he is playing when he is using simple software agents for example for internet searches. Therefore, the human itself is regarded and modelled as an agent: the so-called *human based agent*. The human based agent consists of humans as well as of user interfaces and co-operates with other human based agents, software agents or technical agents within manufacturing processes. The agents themselves are symmetrically structured by agent head and agent body. A Multi Agent System is consisting of agents and the agent platform. The agent platform is the required agent infrastructure for communication control etc.

4 Second Step: Detailed Design and Agent Implementation

In the following, a short overview for the design and implementation with the *AgenTec* project is given. This is based on the FIPA *(Foundation for Intelligent Physical Agents)* specifications [3] because FIPA has been identified as the most suited agent specification. In this analysis it has been shown that FIPA standard originally developed for pure software agent systems can be transferred to manufacturing systems and applications. Developing tools, FIPA compliant agent platforms [8], FIPA Agent Communication Language and several interaction protocols [3] such as FIPA contract net interaction protocol can be applied on production and manufacturing. With regards to the analysis of FIPA and existing physical MAS, some problems have been identified and further requirements, which are different or additional to the requirements of pure software MAS, have been taken into account within *AgenTec*.

The FIPA compliant interaction protocols have to be modified for new application cases. Several protocols, e.g. for mobile transport robots, have been developed. In addition, the Unified Modeling Language (UML) as well as the Agent Unified Modeling Language (AUML) have been used as a semi-formal notation [9]. Furthermore, the ODEMA method (Object oriented Method for Developing Technical Multi Agent Systems), which is developed by the Institute of Industrial Manufacturing and Management, Stuttgart, has been partially used within the analysis and the design phase [16].
Apart from the infrastructure for agent management, i.e. the agent platform, this work was also devoted to develop co-operation processes. Therefore, the mediated package negotiation has been developed (fig. 3).

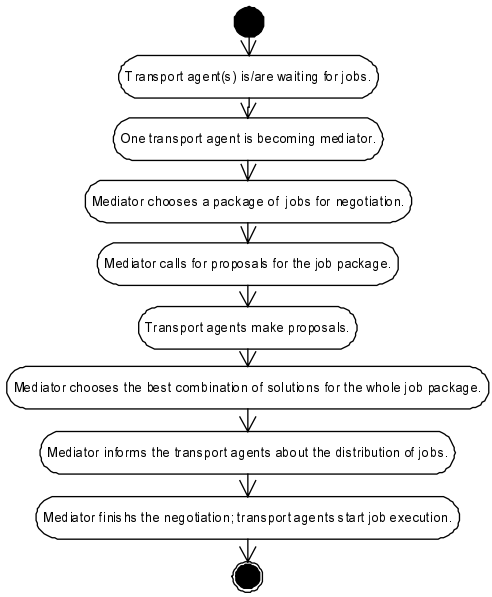


Fig. 3. Mediated Package Negotiation (e.g. on base of mobile transport agents)

The package negotiation is based on a mediator which is initialising a new negotiation phase and which is active participant within the negotiation, i.e. the mediator is making proposals by itself and in the case of winning the competition it executes the corresponding tasks. Furthermore, the role and position of the mediator is given dynamically and not restricted on special agents. In addition, the mediator can use its knowledge of several tasks and offers (proposals) to optimise the distribution of tasks to negotiation participants. The package negotiation is not limited to transport tasks; it can be transferred to other tasks and even to conflicts such as resource conflicts. As a special case, the iterative learning has been considered to optimise the distribution of free, i.e. available, transport agents, on the base of package negotiation [9].
Within *AgenTec* several prototypes of agent platforms have been developed consisting of at least two mobile robots of Care-O-bot type, two industrial robots Stäubli RX90 and two mobile KURT robots. Furthermore, the first generation of

MAS was combined with the FactoryBroker™ [6,10]. Our new agent platform is based on JINI™ because of increasing requirements. Another focus is the development of hybrid robot architectures (reactive, deliberative and planning systems) [13]. The different software applications (services, agent platforms, agents, simulators, virtual reality applications, hybrid robot architectures etc.) are developed with object oriented languages such as C++ or Java.
At the moment we are developing and implementing a physical material flow demonstrator for commission. The scenario is the commission of different coloured pens and cups (fig. 4). The material flow system consists of two automated commission cells (industrial robot), one manual workplace and two different types of mobile transport robots. The cells and mobile robots are modelled as production agents. The worker is supported by a PC for human-machine interaction. All cells consist of magazines for work pieces and storage. The newest generation of *AgenTec* systems is planned to be presented during SPS/IPC/DRIVES 2003 in Nuremberg, Germany.

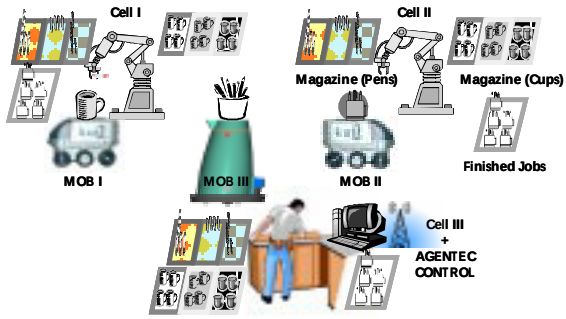


Fig. 4. *AgenTec* scenario

5 Summary and Conclusion

An approach for the development of agent technology – applied within the *AgenTec* project for the development of agent technology - was presented. This approach can be transferred to other technical applications on the area of manufacturing systems. It is based on processes and methods, e.g. for the agentification of systems as agents, on physical and software systems and tools. The integral view of processes, methods, tools, data, experts and systems will lead to *"plug and produce"* and enable developers and other participants of the product development and production development process to improve their work in terms of quality for engineering.

Acknowledgements. *AgenTec* is a joint BMBF (German Federal Ministry of Education and Research)-funded project (grant. no. 01AK905A/B) between the Fraunhofer Institute AIS, St. Augustin, and the Fraunhofer Institute IPA, Stuttgart.

References

1. Albert, M.: DIAMOND reference model. Report, Esprit Project No. 28735, October 2001
2. N.N.: AgenTec – "Agent Technology for Robots in Production and Services", URL: http://www.agentec.de/, December 2002
3. N.N.: FIPA – "Foundation for Intelligent Physical Agents", URL: http://www.fipa.org/, April 2002
4. DeLoach, S.: http://www.cis.ksu.edu/~sdeloach/publications/Conference/MaSE-maics 20001.pdf, Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001). Miami University, Oxford, Ohio, March 31 - April 1, 2001
5. Langer, G.: HoMuCS – A methodology and architecture for Holonic Multi-cell Control Systems. Ph.D. thesis, Technical University of Denmark, 1999
6. Neubert, R., Colombo, W., Ax, U., Schoop, R.: An Approach to Integrate a Multiagent-based Production Controller into a Holonic Enterprise Platform. Proc. 1st Int.l Conf. on Information Technology in Mechatronics (ITM'01), Special Session, Istanbul, Turkey, October 1–3 2001, 65–70
7. N.N. : Plant automation based on distributed systems. URL: http://www.pabadis.org/htdocs/home.html, status: May 2003
8. Petsch, M.: Aktuelle Entwicklungsumgebungen für mobile Agenten und Multi-agenten-systeme. Wirtschaftsinformatik 43, Nr. 2 (2001), pp. 175–182 (in German)
9. Ritter, A.: A Multi Agent System for Mobile Systems in Production Systems, Ph.D., Jost Jetter, 2003 (in German)
10. Ritter, A., Braatz, A., Höpf, M.: Models and Modules for Production Agents. Proceedings of the 1st International Conference on Information Technology in Mechatronics (ITM'01), Special Session, Istanbul, Turkey, October 1-3 2001, pp. 47–52
11. Ritter, A., Baum, W., Höpf, M., Westkämper, E.: Agentification for Production Systems. Proc. INT 2002: Second Intl. Workshop on Integration of Specification Techniques for Applications in Engineering, Grenoble, France, April 6-7 2002, 21–28
12. Ritter, A., Richter, H., Westkämper, E.: Agent Life Cycle Management for Manufacturing Systems, CIRP 36th International Seminar on Manufacturing Systems, Saarbrücken, Germany, June 3–5 2003
13. Schönherr, F., Hertzberg, J.: The DD&P Robot Control Architecture - A preliminary report, Advances in Plan-Based Control of Robotic Agents. Springer Lecture Notes in Artificial Intelligence, Volume 2466, (M. Beetz, J. Hertzberg, M. Ghallab, M. Pollack, eds.), 2002, 249–269
14. Sugi, M., Maeda, Y., Aiyama, Y., Arai, T.: A Flexible Robot System for Assembly with a Concept of Holon. Proceedings of the 32nd ISR (International Symposium on Robotics), vol. 2, 19.-21. April 2001, pp.844–849
15. N.N.: SFB 467 – Transformable Enterprise Structures, URL: http://www.sfb467.uni-stuttgart.de/, July 2002
16. Westkämper, E., Braatz, A.: Eine Methode zur objektorientierten Spezifikation von dezentralen Automatisierungssystemen mit der Unified Modelling Language (UML), at Automatisierungstechnik, Heft 5 (2001), 225–233 (in German)

Cost-Based Dynamic Reconfiguration System for Evolving Holarchies

Francisco P. Maturana[1], Pavel Tichý[2], Petr Šlechta[2], Raymond J. Staron[1], Fred M. Discenzo[1], Kenwood Hall[1], and Vladimír Mařík[2]

[1] Rockwell Automation, 1 Allen-Bradley Drive, Mayfield Hts., OH 44124-6118, USA
[2] Rockwell Automation Research Center, Americka 22, 120 00 Prague, Czech Republic
{fpmaturana, ptichy, pslechta, rjstaron, fmdiscenzo, khhall, vmarik}@ra.rockwell.com

Abstract. Autonomous, highly distributed control architectures are composed of a significant number of holons/agents that can reason and act on behalf of represented processes or artifacts in a coordinated manner. Depending on the social organization capabilities of the agents, the autonomous system could evolve into complex agent organizations called temporal holarchies. Cost-based negotiation supports the holarchy formation. Dynamic hierarchical teamworks architecture of middle-agents is described to increase robustness of the architecture.

Keywords: Adaptation, Agents, Distributed, Cost, Intelligence, and Reconfiguration.

1 Introduction

A requirement for new automation systems is increased survivability of the control system. To fulfill this, we use a highly distributed control architecture, where the automation devices are extended to enable the creation of agents. In this paper, holons and agents have similar meaning. Additional definitions and interpretations of holon and agent technologies can be found in [1], [2], [3], [4], and [5].

In this paper, temporal holarchies are understood as task-centric cost-based organizations. Dynamic creation of temporal holarchies is supported by middle-agents that are a part of the multi-agent system. Middle-agents are agents that help others to locate and connect to agent providers of services [11]. Middle-agents are used by holons/agents to locate others based on a selected capability and to form more complex sets of agents or holarchies. A dynamic hierarchical teamwork of middle-agents is used to increase the robustness of the architecture. The holarchies form and evolve toward convergence using cost-models.

The focus of this work is on methods for the creation of highly distributed control agents. The holon/agent architecture is organized according to the following characteristics:

- Autonomy: Each agent makes its own decisions and is responsible for carrying out its decisions toward successful completion.
- Cooperation: Agents combine their capabilities into collaboration groups (clusters or blocks) to adapt and respond to diverse events and mission goals.
- Communication: Agents share a common language to enable the cooperation.

V. Mařík, D. McFarlane, P. Valckenaers (Eds.): HoloMAS 2003, LNAI 2744, pp. 310–320, 2003.

- Fault tolerance: Agents possess the capability to detect equipment failures and to isolate failures from propagating [8].
- Proactive behavior: Agents periodically or asynchronously propose strategies to enhance the system performance or to prevent the system from harmful states.

Agents can dynamically change their objectives, control strategy, or operation based on changing local and global goals. Agent planning is carried out in three main phases: Creation, Commitment, and Execution. During creation, an agent initiates a collaborative decision making process. The agents offer a solution for a specific part of the request. Then, the agents commit their resources to achieve the task in the future. Finally, the agents execute the plans.

The agents emit messages inside the organization using a Job Description Language (JDL) [6] and [7] and also outside of the organization via wrapping JDL messages inside a universally accepted communication language. Presently, a commonly accepted language is the Foundation for Intelligent Physical Agents (FIPA) Agent Communication Language (ACL) [10].

An agent exhibits goal-oriented social behaviors to be autonomous, cooperative problem-solving entities. The agents can evaluate both local and group goals. For example, in the chilled water system, when an agent detects a water leakage in a pipe section, it establishes a local goal consisting of stopping the leakage. The agent emits messages to other agents to isolate the damage. This is a group-based decision.

2 Dynamic Hierarchical Teamwork Architecture

During initialization time, each application-specific agent emits a registration message to its local middle-agent (M-A). Then, the local M-A registers the new agent in a database that is later used to match service requests with the agent capabilities. The directory facilitator agent specified by FIPA [10] is used as the M-A.

To increase the robustness of the system, the architecture allows for the creation of multiple M-As. The use of M-As extends the flat architecture to support the agent coordination under a highly flexible management perspective. The hierarchical organization of M-As does not require a single root to serve as a global M-A. Critical considerations are taken into account when designing the M-A system. Since a single global M-A may become a single-point-of-failure and possibly a communication bottleneck, it is important to incorporate organizational capabilities (survival skills) into the design of M-As. These 'social' skills allow the M-A system to survive eventualities such as losing global M-As.

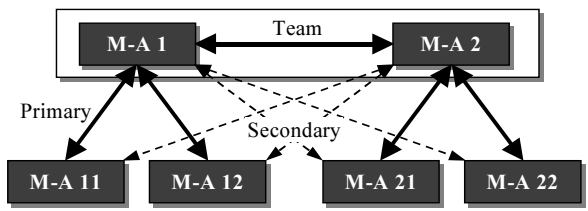


Fig. 1. Example of 2-level dynamic hierarchical teamwork architecture

The user defines an initial structure of M-As at design time via primary communication channels among M-As (solid arrows in Fig. 1), but the system changes its form using secondary channels (dashed arrows) according to need. Primary and secondary channels are physically the same. M-As prefer to use the primary channels over the secondary ones as further described.

If a global M-A has at least two child middle-agents, the global M-A manages communication among them. Because the architecture does not allow direct communication among the child M-As, a failure in the global M-A can cause a split of the whole structure into islands. This separation obstructs the communication among the child M-As since the agent-to-agent relationship network has been broken. In this case, the child M-As cannot find each other. Similarly, any other M-A that is not in a leaf position in the graph can cause the split of respective child M-As.

To provide additional robustness, there has to be a capability in the system to create multiple global M-As. In this case, all global M-As are interconnected and they share the same information. The global M-As form a team (see Fig. 1). When a M-A fails, another M-A compensates for the missing one by handling the request from the child M-As.

During a normal operation, M-As use primary communication channels. Nevertheless, M-As access secondary channels to compensate for the missing primary channels. In this context, a secondary communication channel does not mean 'second'; there can be more than one secondary communication channel per one primary channel. When a M-A decides to use a secondary communication channel, the secondary becomes the primary one. Then, the structure of interconnections of primary communication channels is dynamically reshaped. The architecture can be structured to $N > 2$ agent levels. In this case, teams are organized in hierarchical structures using a tree-like shape. This means that the structure of teams does not contain loops and the graph is connected. This holds only in cases when the primary connections are the edges of the graph. The structure of teams creates a rooted tree, if we select the top-most team as a root.

To detect the failure of a M-A, a heartbeat mechanism is used. M-As form observer-subject pairs. Observers initiate the heartbeat action with subject agents. The subject agents then periodically send their status to the observer agent. The observer agent internally sets deadlines to receive acknowledgements from the subject agents. In case that a subject agent fails to send a heartbeat signal, the observer agent dismisses the failed agent from the active team. The observer agent carries out reorganization and propagates the information about the failed agent.

3 System Modeling

Another relevant aspect of this architecture is the criteria to create agents: How large or how small can an agent be? The complexity and dynamics of the application makes it difficult to establish the agents' sizes and spheres of use. To establish an agent boundary, we consider the proximity of the devices and their relationships. There are cases in which it makes more sense to model an agent as a standalone device. In other

cases, it makes more sense to group the devices under one agent representative. These decisions have a direct effect on the size and complexity of the agent organization.

We use a Chilled Water System (CWS) pilot facility, which includes automation technology for control and visualization. Fig. 2 shows square boxes representing heat loads and water services. A water service provides a load with cold water. Immersion heaters provide stimuli for each service to model heat transfer. There are 3 subsystems, plants, mains, and services. There is one plant per zone (2 zones).

The water-cooling plants (ACP) are modeled as single agents. The main circulation piping is partitioned among 'T' pipe sections, which are also agents. Load agents (SVC) include a heat generator and a temperature sensor. There are standalone valves in the main circulation loop for the supply and return lines represented by valve agents (V). The total system has 68 agents.

Each agent is associated with capabilities and each capability is associated with specific set of operations. The negotiation among agents uses local planning and negotiated planning (i.e. cooperation). The agents use their local 'world observations' to determine their actions which are translated into execution steps. In the negotiated planning, the agents discover each other's capabilities.

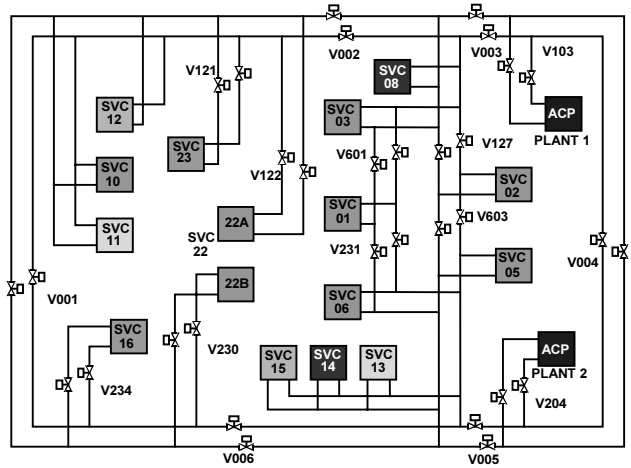


Fig. 2. Chilled Water System

A chilled water system action is expressed as a concatenation of partial actions. For example, when a Load agent detects that it will soon overheat due to its current heating trend, it sets an event for controlling the overheating phenomenon within a specific time. Since a Load agent does not have a cooling capability, it makes a cooling request (e.g., SupplyCooling). The middle-agents provide the addresses of the agents that support such a capability. ACP agents are contacted to satisfy the request. However, since these agents only know how to supply water, they need to propagate the request under a different context. At this time, the initial request has changed form several times. The request is sent to the Valve agents, which later initiate a request for water regulators to transmit the water to the load.

Other factors like the cost and state of the equipment influence the agents' observations (beliefs). Such factors help the agent to decide the type of operation to be performed. A Water Service agent has the capability to decide among three operations: 1) canSupplyCooling, 2) startCooling, and 3) stopCooling.

The agents are distributed according to the component's physical location. The hardware partitioning follows the wiring configuration of the input/output (I/O) signals in such a way that the components are independent of one another. Single-point-of-failure nodes, from the hardware distribution perspective, are avoided. Other rules need to be satisfied to be free from single-point-of-failure nodes. Example rules are: No remote I/O, one agent per component and controller, no mapped inter-controller data, etc. The facility has 20 FlexLogix 5434 controllers for the Services and Loads and 3 Logix 5550 controllers for the Process-level entities and ACP plants.

4 Cost-Based Evolving Holarchies

The cost model is based on the ContractNet protocol [9] that specifies the relationship between bidders and bid-evaluators. Groups of bidders and bid-evaluators form blocks of negotiators. Several blocks of negotiators emerge from any of the agents (bidders and evaluators) depending on the complexity an extension of the tasks.

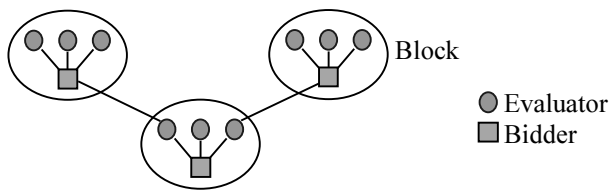


Fig. 3. Negotiation blocks

Fig. 3 shows a snapshot of three blocks of negotiators. The aggregation of blocks forms the basis of a holarchy. The first block is created in response to a task request that is initiated by an agent or a human. This request arrives to an agent as a request for supplying a specific service. In our application, a Load agent that represents an overheating component emits a typical request. The providers of cold water (ACP plants) respond to this request. Thus, in this initial transaction, we see the formation of a first negotiation block. The Load agent is a bidder and the providers of water are bid-evaluators. The next step is to determine the availability of the service. This implies several inter-locutions between the agents, in which the bid-evaluators assess the availability of the service and its corresponding nominal cost. The nominal cost is simply a cost per unit of provided service. The local cost may also be augmented by other dynamic factors that currently affect the operations of the service provider. For example, the ACP plant may be reaching its maximum efficiency. Thus providing additional cooling at this point in time means higher cost of operation.

The notion of cost is associated with cost of operation that an agent will charge to carry out a control action. Each agent's cost is a user-defined parameter. The bidder

initiates the request by including an initial quantity to be paid for the service. The initial quantity is a user-defined parameter. By combining both situations corresponding to the cost calculation and purchasing power, the bidder agent learns its association with a bid-evaluator agent. In our example, after the evaluation of the bid is complete, the Load agent learns about its association with two suppliers. Consequently, the bidder agent selects the least expensive result (assuming that the sealed bid auction is used). This agent interaction provides a mechanism to discover agent-to-agent relationships, which is a type of learning behavior.

A conclusion from the above is a set of agent behaviors that needs to be given to the agent infrastructure to enable simple negotiations. In general terms, the mechanism can be summarized in the following steps: (a) task emergence, (b) sub-tasking, (c) agent discovery, (d) negotiation, and (e) decision. Each step above requires additional control, communication, and artificial intelligence technologies. Fig. 4 shows the schema that associates the above steps into a template protocol. This protocol facilitates the formation of a single level holarchy.

The first negotiation block is created in response to a task request that is initiated by an agent or a human through an operator interface. This request arrives to an agent as a request for supplying a specific service. In our application, a Load agent that represents an overheating component emits a typical request. The providers of cold water (ACP plants) respond to this request. Thus, in this initial transaction, we see the formation of a first negotiation block. According to the ContractNet protocol, the Load agent is a bidder and the providers of cold water are bid-evaluators.

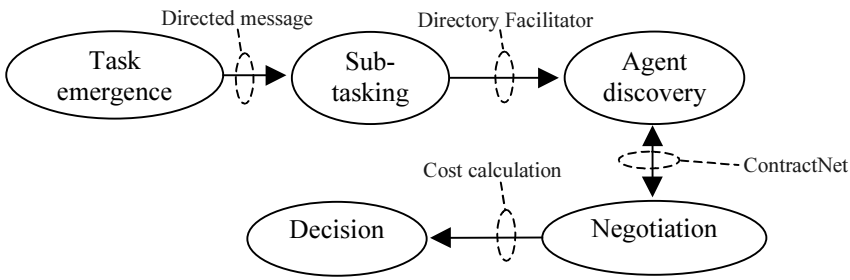


Fig. 4. Template Protocol for Single-level Holarchy

Any agent can initiate a new request outside of its base negotiation block. A similar mechanism applies in this case, where the agent metamorphoses from bid-evaluator into a bidder, dynamically changing its role in the negotiation process. This is another very important quality of autonomous agents. The agent needs to halt its current planning until external resources fulfill an intermediate part. This is a partial planning action that is used to expand the single-level holarchy.

Since each negotiation block has a finite amount of time to come up with a result, the newly formed negotiations must subordinate their life cycles to the time limitations of their predecessor negotiation blocks. In this manner, the expansion of the holarchy is auto-regulated to restrain any of the branches from escaping into expensive search regions. In general terms, we use time to influence the unfolding of

the search tree. This is a time-based branch-and-bound technique. Although the M-A system helps to accelerate the discovery of agents that fulfill a specific requirement, the concatenation of several negotiation blocks increases the decision latency. Without a M-A system, the decision latency would be inadequate for the automation/control domain under consideration.

The holarchy scales up depending on the agents' attributes to respond to a task, task decomposition, the cost involved in the negotiation, and the time required for the completion of the solution. If one intention of the agent system is to minimize the cost of operation, the agents tend to select low cost associations instead of high cost associations. This criterion does not guarantee an optimal solution since the time constraint limits the possibility to search all possible combinations of solutions and costs. Hence, possibly, the agents are able to build solutions that satisfy each agent's physical limitation at a given available low cost.

As the planning time shrinks, the solutions remain more feasible but less optimal. It must be noted that in time-critical applications, cost calculations must be tightly coupled to the relative times associated with the actual control. In our CWS example, we observe the expansion of the single-level holarchy when the cold water providers decide to break the cold-water request into two subtasks: 1) find a water route to supply water to the load and 2) find a water route to return the water from the Load.

At this point, the initial negotiation block (cold water negotiation) has been broken down into subblocks. After the subtasking, each bid-evaluator is transformed into a bidder for path creation and the negotiation block is halted until a response comes back from the subblocks. In the sub blocks, the template protocol applies in the same manner. In order to converge to a feasible and affordable solution, the agents carry out the cost calculation pertinent to the level and the context of the negotiation, which in this case is water transportation cost.

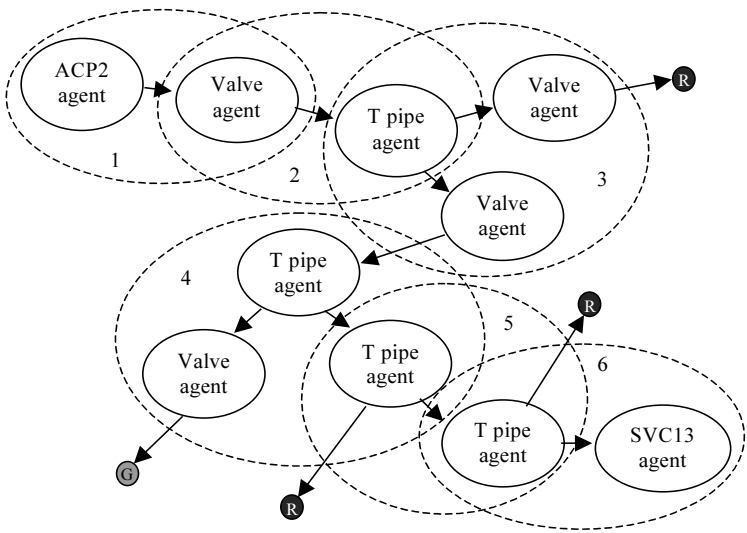


Fig. 5. Evolved Holarchy

Fig. 5 shows an expanded holarchy for a water route planning to transport water from ACP2 into the load attached to SVC13. The figure only shows six negotiation blocks, where a combination of different agents carry out the planning process for a successful route. The *R* branches represent terminal nodes where the branches will not expand during the current planning cycle due to some higher order constraints or high aggregated cost. For instance, water from one ACP plant cannot be mixed with the water from another ACP plant. In such a case, a valve agent that may break the segregation of water may acquire very high cost of operation.

The system auto regulates itself using the notion of cost but also using the process restrictions. The letter *G* represents a feasible branch that may reach the load. However, this branch is not chosen due to its high aggregated costs. Since the agents operate in a highly distributed, parallel concurrent decision making system, the feasibility of the branches is kept available until the sub-level blocks decide if the branch is viable or not. This promissory availability of resources requires a robust coordination engine that synchronizes the operation of the agents with the real time events emerging from the agent interaction. The latencies within distributed communication networks affect the decision cycles, forcing the agents to adjust their local expectations for answers. An agent, upon failing to create a route, may decide to alter its purchasing power to enable other branches to enter into the decisions. How much can an agent vary these hard constraints is a user decision.

Upon successfully creating a route, the holarchy contracts back to its originator block. In each negotiation block, the agents complete partial decisions by selecting the least cost solutions. In this retraction process, it may also occur that the agents branch out to perform other subtasks, therefore halting the block momentarily. It will be difficult to program all possible combinations of decision branches ahead of time.

Since these dynamic occurrences are difficult to forecast, it has been learned that it is more practical to program the agent system with simpler rules and behaviors to support the agent negotiation in dynamically evolving organizations instead of attempting a full scale centralized solution.

The holarchy contracts to its initial level where everything started from a request for cold water. The initial block that has been waiting for the answers from the subtasks is restarted and the agents decide upon a plan. This process concludes with a global plan to provide cold water to the load. With the global plan established, resources are committed and plan execution is initiated. During the commitment phase, each agent involved in the successful plan is notified to prepare its equipment for the execution phase. Each agent maintains a piece of the global plan to support future requests for their services. For example, a valve agent knows both sources and consumers of water. This knowledge helps the valve agent decide its participation in future plans and the cost associated with it. There are cases in which a valve agent helps the water circulation from one ACP plant. The valve agent rejects request for circulation of the water from a second ACP plant. This is one example of using the knowledge that was obtained from the holarchy for subsequent decision making.

5 Results

We built a physical prototype of a chilled water system to test the capability of the agent system to reorganize its components based on two conditions:

1. reorganization after failure of the structure of middle-agents and
2. organization of the system's capabilities upon a request condition using cost-based evolving holarchies.

Fig. 6 shows the communication traffic of a small collection of agents. The test is split into time slots of 20 seconds (X-axis) and values (Z-axis) represent the number of messages per slot. The graph GM-A1 (Y-axis) considers outgoing messages from the first global middle-agent, GM-A2 shows outgoing messages from the second middle-agent, and the graph GM-A3 shows the last global middle-agent.

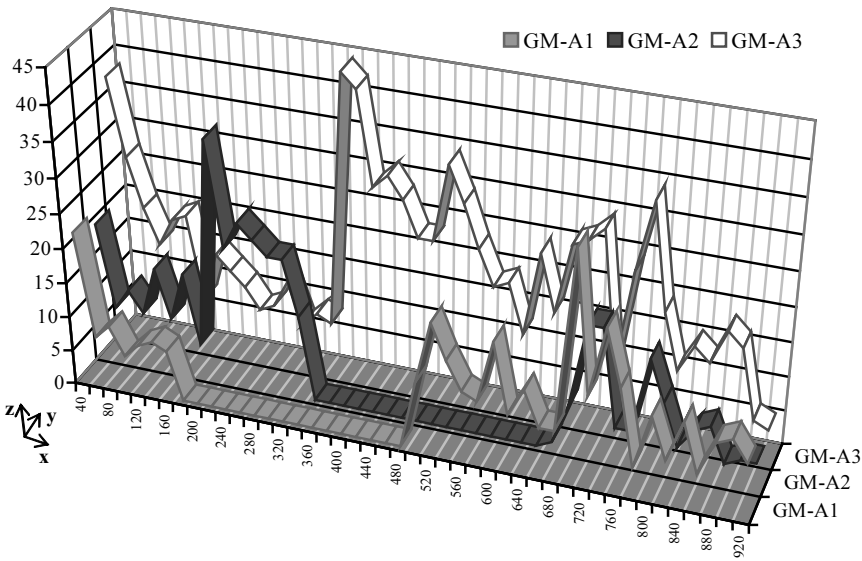


Fig. 6. No response failures of global middle-agents

After 150 seconds, the no response failure at the first global middle-agent was simulated. The system stopped using the first global middle-agent and further requests were set pending until the agents discovered the malfunctioning. The agents changed over to the second middle-agent. Fig. 6 also shows the switch to the second middle-agent when the communication traffic increased from 10 messages to 45 messages approximately at the third global middle-agent.

At 300 seconds, another event was introduced to fail the second middle-agent. The agents changed over to the third middle-agent. After 450 seconds, the first global middle-agent was repaired followed by the second middle-agent 150 seconds later. The local middle-agents changed back to communicating with their initial middle-agents and the system recovered its original configuration. Thus, the robustness criterion was demonstrated in this simulation study.

Fig. 7. Water Routing Decision Tree

Fig. 7 shows the decision tree for a water route plan to move water from ACP01 plant to SVC22S service. The light gray lines and boxes indicate successful paths. The successful paths are evolving holarchies as described in Section 5. The holarchies for planning appear spontaneously and remain active for short periods. After the negotiation is complete, each agent executes the plan and keeps their partial plans.

6 Conclusion

This paper describes the criteria to establish system functions and dynamically emerging relationships among the agents without preprogramming the relationships. The criteria are demonstrated with a chilled water application and robustness has been demonstrated with a simulation study. The cost-based holarchy appear to be robust and effective technique to support agent negotiations and dynamic reconfiguration.

References

1. Brooks, R.A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, 2(1) (1986) 14–23
2. Christensen, J.H.: Holonic Manufacturing Systems: Initial Architecture and Standards Directions. First European Conference on Holonic Manufacturing Systems, Hanover, Germany (1994)
3. Wooldridge, M. and Jennings, N.: Agents: Theory and Practice. Knowledge Engineering Review, Vol. 10, No. 2 (1995) 115–152
4. Shen, W., Norrie, D., and Barthès, J.P.: Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. Taylor & Francis, London (2001)
5. Mařík, V., Pěchouček, M., and Štěpánková, O.: Social Knowledge in Multi-Agent Systems. In: Multi-Agent Systems and Applications (Luck M., Mařík V., Štěpánková O., Trappl R., eds.) LNAI 2086, Springer, Berlin (2001) 211–245
6. Maturana, F., Staron, R., Tichý, P., and Šlechta, P.: Autonomous Agent Architecture for Industrial Distributed Control. 56th Meeting of the Society for Machinery Failure Prevention Technology, Sec. 1A, Virginia Beach (2002)

7. Chiu, S., Provan, G., Yi-Liang, C., Maturana, F., Balasubramanian, S., Staron, R., and Vasko, D.: Shipboard System Diagnostics and Reconfiguration using Model-based Autonomous Cooperative Agents. ASNE/NAVSEA Intelligent Ship Symposium IV, Philadelphia, PA (2001)
8. Discenzo, F.M., Rusnak, D., Hanson, L., Chung, D., and Zevchek, J.: Next Generation Pump Systems Enable New Opportunities for Asset Management and Economic Optimization. Fluid Handling Systems, Vol. 5, No. 3 (2002) 35–42
9. Smith, R.G.: The Contract Net Protocol. High-level Communication and Control in a Distributed Problem Solver. In IEEE Transactions on Computers, C-29(12) (1980) 1104–1113
10. The Foundation for Intelligent Physical Agents (FIPA): http://www.fipa.org
11. Klusch, M. and Sycara, K.: Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In (Omicini A. et al., eds.) Coordination of Internet Agents, Springer-Verlag, Berlin (2001) 197–224

Author Index