Vladimír Mařík
Robert W. Brennan
Michal Pěchouček (Eds.)

LNAI 3593

Holonic and Multi-Agent Systems for Manufacturing

Second International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2005
Copenhagen, Denmark, August 2005, Proceedings

Springer

Lecture Notes in Artificial Intelligence 3593

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Vladimír Mařík Robert W. Brennan
Michal Pěchouček (Eds.)

Holonic and Multi-Agent Systems for Manufacturing

Second International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2005
Copenhagen, Denmark, August 22-24, 2005
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, Department of Cybernetics Technická 2
16627 Prague 6, Czech Republic
and Rockwell Automation Research Center
Prague Pekařská 10a/695, 15500 Prague 5, Czech Republic
E-mail: marik@labe.felk.cvut.cz

Robert W. Brennan
University of Calgary
Department of Mechanical and Manufacturing Engineering
2500 University Drive NW, Calgary, Alberta T2N 1N4, Canada
E-mail: rbrennan@ucalgary.ca

Michal Pěchouček
Czech Technical University
Faculty of Electrical Engineering, Department of Cybernetics
Technická 2, 16627 Prague 6, Czech Republic
E-mail: pechouc@labe.felk.cvut.cz

Library of Congress Control Number: Applied for

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

ISSN 0302-9743
ISBN-10 3-540-28237-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-28237-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11537847 06/3142 5 4 3 2 1 0

Preface

The challenge faced in today's manufacturing and business environments is the question of how to satisfy increasingly stringent customer requirements while managing growing system complexity. For example, customers expect high-quality, customizable, low-cost products that can be delivered quickly. The systems that deliver these expectations are by nature distributed, concurrent, and stochastic, and, as a result, increasingly difficult to manage. Unfortunately, the traditional hierarchical, strictly centralized approach to control used in these domains is characteristically inflexible, fragile, and difficult to maintain.

These shortcomings have led to the development of a new class of manufacturing and supply-chain decision-making approaches in recent years. Solutions based on these approaches usually explore a set of highly distributed decision-making units that are capable of autonomous operations while cooperating interactively to resolve larger problems. The units, referred to as *agents* in classical computer science and software engineering, or *holons* if physically integrated with the manufacturing hardware, interact by exchanging information. These units are motivated by arriving at local solutions as well as collaborating and sharing resources and goals in solving the overall problem in question collectively.

Despite a focus on the manufacturing domain in the late 1990s, applications of the holonic and agent-based approach were not restricted to this area and currently span a wide range of applications such as highly distributed real-time control systems, discrete event simulation, RFID technology, business and supply-chain management, etc. The research communities working in these different fields approach the problem of intelligent industrial solutions from different viewpoints and have already started to cooperate efficiently. Thus, global visions of applying the agent-oriented design philosophy from the level of real-time highly distributed execution control to dynamic operational planning on the machine or workshop up to managerial tasks connected with running the businesses have appeared. We can see solutions requiring coalition formation and teamwork planning, strongly supported by the challenging visions of virtual manufacturing and virtual enterprises. There is evident convergence in terminology, standards and methods. Moreover, we can clearly document that this convergence is amplified and catalyzed by the requirements from the real industrial *environment.*

In recent years we can identify trends toward *feasibility validation* of agent-based solutions, usually exploring specialized agent-based simulation tools. *Agent-based simulations* together with the first considerations on how to leverage the significant progress in the *RFID technology* field for discrete manufacturing represent dominant trends in the current industry-oriented research of agent-based systems.

This strong multi-agent community, organized around groups such as FIPA (Foundation for Intelligent Physical Agents) or AgentLink (European Co-ordination Action for Agent-Based Computing), is aware of the fact that one of the most challenging areas for the application of the agent-based computing and decision-

making systems is the *field of intelligent manufacturing*. Special activities focused at industrial applications of agent-based computing have been established recently. For example, the AgentLink Agent Technology Conference annually brings together commercial organizations interested in exploiting agent technologies to benefit their businesses. As well, an Industry Track was organized for the first time at AAMAS, the leading international conference in the field of multi-agent systems. This track brings together researchers and academics on one side and commercial developers and key technology decision makers on the other side.

We are convinced of the value and importance of the continuation of the HoloMAS events. In particular, HoloMAS was among the first pioneering melting pots for ideas connected with distributed decision-making and control and has already gained international reputation. The first four HoloMAS events held under the DEXA event umbrella (three workshops, particularly HoloMAS 2000 in Greenwich, HoloMAS 2001 in Munich and HoloMAS 2002 in Aix-en-Provence as well as the first HoloMAS 2003 conference held in Prague) helped to bring together the research communities focused on agent-based industrial solutions, to realize the joint principles of agent-oriented applications on different levels of manufacturing, factory and supply chain management and to integrate better their research activities and results. At HoloMAS 2005 we wanted to document the feasibility and viability of the initial ideas, to show the continuity of the industrial agent-oriented research and to make the progress in the field clearly visible.

We expected that the HoloMAS 2005 conference would create an excellent, highly motivating environment, and help to continue to integrate the community. It was expected to contribute to a clarification of the goals and to a more efficient coordination of the research in the subject fields. This conference also continued to serve as a window to current holonic and agent-based manufacturing research and, as such, offered information about the state of the art to specialists in neighboring, knowledge-processing research fields covered by the DEXA multi-conference event. We are very thankful to the DEXA Association for providing us with this excellent opportunity.

For this year's edition, 40 high-quality papers were submitted by the most important, core research bodies engaged in holonic and agent-based manufacturing worldwide. After a careful reviewing process the Program committee selected 23 papers to be presented and included in this volume. They contain the most representative results of the corresponding research and provide an excellent overview of what is the current state of the art.

We would like to thank also both the AgentLink III EU Coordinated Action and the I*PROMS EU Network of Excellence for their support and technical co-sponsorship.

Prague, Calgary
June 2005

Vladimír Mařík
Robert W. Brennan
Michal Pěchouček

HoloMAS 2005

2nd International Conference on Industrial Applications of Holonic and Multi-agent Systems (HoloMAS 2005)

Applications of Holonic and Multi-agent Systems

Copenhagen, Denmark, August 22–24, 2005

Program Co-chairs

Vladimír Mařík	Czech Technical University in Prague, and Rockwell Automation, Czech Republic
Robert W. Brennan	University of Calgary, Canada
Michal Pěchouček	Czech Technical University in Prague, Czech Republic

Program Committee

Jose Barata	Universidade Nova de Lisboa, Portugal
Vicente Botti	Universidad Politecnica de Valencia, Spain
Jeffrey M. Bradshaw	University of West Florida, USA
Monique Calisti	Whitestein Technologies, Switzerland
Luis Camarinha-Matos	Universidade Nova de Lisboa, Portugal
Armando W. Colombo	Schneider Group, France
Misbah Deen	University of Keele, UK
Thomas Strasser	Profactor, Austria
Klaus Fischer	DFKI GmbH, Germany
Martyn Fletcher	Agent Oriented Software Ltd., UK
William Gruver	Simon Fraser University, Canada
Kenwood H. Hall	Rockwell Automation, USA
Matthias Klusch	DFKI GmbH, Germany
Dilip Kotak	National Research Council of Canada, Canada
Jiří Lažanský	Czech Technical University in Prague, Czech Republic
Francisco P. Maturana	Rockwell Automation, USA
Duncan McFarlane	Cambridge University, UK
Gerard Morel	CRAN, France
Joerg Mueller	Siemens AG, Germany
Douglas Norrie	University of Calgary, Canada
Gregory Provan	University Cork College, Ireland
Leonid Sheremetov	Mexican Oil Institute, Mexico
Alexander Smirnov	SPIIRAS, Russia
Shinsuke Tamura	Fukui University, Japan

Ambalavanar Tharumarajah	CSIRO, Australia
Paul Valckenaers	Katholieke Universiteit Leuven, Belgium
Hendrik Van Brussel	Katholieke Universiteit Leuven, Belgium
Edwin H. Van Leeuwen	BHP-Billiton, Australia
Tomáš Vlček	Czech Technical University in Prague, Czech Republic
Pavel Vrba	Rockwell Automation, Czech Republic
Valeriy Vyatkin	Martin Luther University, Halle, Germany
Peer-Oliver Woelk	IFW, University of Hannover, Germany

External Reviewers

Jose L. Martinez Lastra	Tampere University of Technology, Finland
Paulo Leitão	Polytechnic Institute of Bragança, Portugal
Pavel Tichý	Rockwell Automation, Czech Republic
Jiří Vokřínek	Czech Technical University in Prague, Czech Republic

Organizing Committee

Pavel Jisl	Czech Technical University in Prague, Czech Republic
Zuzana Hochmeisterová	Czech Technical University in Prague, Czech Republic
Hana Krautwurmová	Czech Technical University in Prague, Czech Republic
Jiří Lažanský	Czech Technical University in Prague, Czech Republic
Vladimír Mařík	Czech Technical University in Prague, and Rockwell Automation, Czech Republic
Gabriela Wagner	FAW, University of Linz, Austria

Table of Contents

Invited Papers

Theoretical and Methodological Issues

Algorithms and Technologies

Implementation and Validation Aspects

Applications

Supply Chain Management

Experience with Holonic and Agent-Based Control Systems and Their Adoption by Industry

Kenwood H. Hall[1], Raymond J. Staron[1], and Pavel Vrba[2]

[1] Rockwell Automation, Inc., 1 Allen-Bradley Drive,
Mayfield Heights, OH, 44124 USA
{khhall, rjstaron}@ra.rockwell.com
[2] Rockwell Automation s.r.o., Research Center Prague, Pekarska 695/10a,
155 00 Prague 5, Czech Republic
pvrba@ra.rockwell.com

Abstract. Currently industrial automation systems are built using a hierarchical top-down approach, yielding tightly coupled and low flexibility systems. Holonic and intelligent agent-based industrial control systems have the potential to be much more highly robust and flexible systems with very loose coupling between subsystems. Despite this potential these systems are slow to be adopted by industry. This paper explores Rockwell Automation's current agent philosophy, application experience, the obstacles to widespread adoption of agent technology in industrial automation systems, and its recent activities to overcome some of the obstacles.

1 Agent Philosophy

Holonic manufacturing systems (HMS) represent a novel paradigm to address some critical problems faced by manufacturing businesses in the twenty-first century. Ever increasing customer requirements are calling for new manufacturing strategies satisfying the needs for (i) open and dynamic structures to allow the on-line integration of new subsystems or removal of existing subsystems from the system without the need to stop and reinitialize the working environment, (ii) agility to adapt quickly to continuous and unanticipated changes in the manufacturing environment, and (iii) fault tolerance to detect and recover from a failure by minimizing its impact on the whole system.

Distributed intelligent manufacturing can meet these requirements. The more traditional sequential and centralized solutions, used within the scope of such agile environments, do not work since they are slow to react, impose operational bottlenecks and are a critical point of failure. Holonics is a decentralized 'bottom up' approach and provides principles to ensure a higher degree of responsiveness and handling of system complexity. The fundamental building blocks of a HMS are called *holons*, fundamentally as presented in [1], to reflect the fact that these entities: (i) are both parts and wholes and (ii) behave simultaneously in an autonomous and cooperative fashion.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 1 – 10, 2005.

The vision of a *holonic factory*, in which all the operations (including product ordering, planning, scheduling, manufacturing, and invoicing the customer) are based entirely on holonic principles, covers several levels of information processing for manufacturing. At least three separate levels can be distinguished:

- **real-time control**, tightly connected with the physical level of manufacturing equipment
- **production planning and scheduling** both on the workshop and factory level
- **supply chain management,** integrating the particular plant with its external entities (suppliers, customers, partners, sales network, etc.)

The particular research results in the holonic field are connected mainly with real-time control. In the other two subfields, the research centers on the philosophical or architectural level, but the particular implementations exclusively use the research results of multiagent systems (MAS). The HMS community has fully realized that the *function block* based real-time control (utilizing the IEC 61499 standard described, for instance, in [2]) is applicable to control tasks only, i.e., they are not the best way to implement the higher level reasoning of agents. Thus it is necessary to leverage the results achieved in the MAS field to widely exploit the visions of a holonic factory. Several general architectures for combining both the function block and MAS technologies have been designed. The most popular new holon model encapsulates one or more function block oriented devices into a wrapper containing a higher-level software component (see Figure 1).

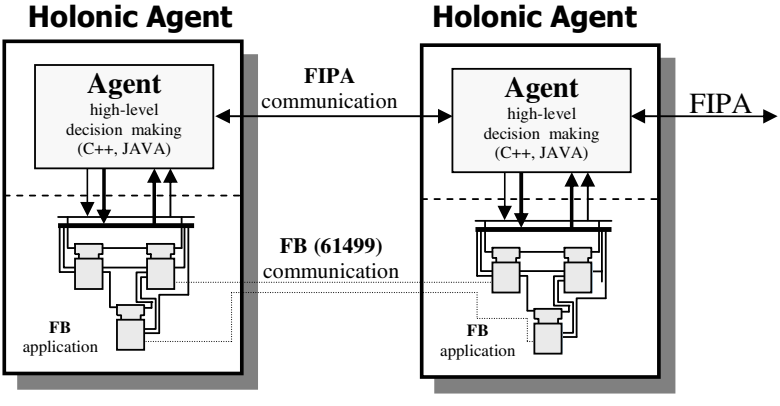


Fig. 1. Model of a PLC-based automation controller with holonic agents using 61499 function blocks for real-time control

Rockwell Automation (RA) has realized that the 61499 function blocks are not as ubiquitous as the IEC 1131 programmable controller languages described in [3], and therefore has implemented its multiagent system using standard relay ladder logic (one of the 1131 languages) instead of 61499 function blocks. The RA model for its *holonic agents* (or simply, *agents*) is still one containing a higher-level intelligent software component, as shown in Figure 2.

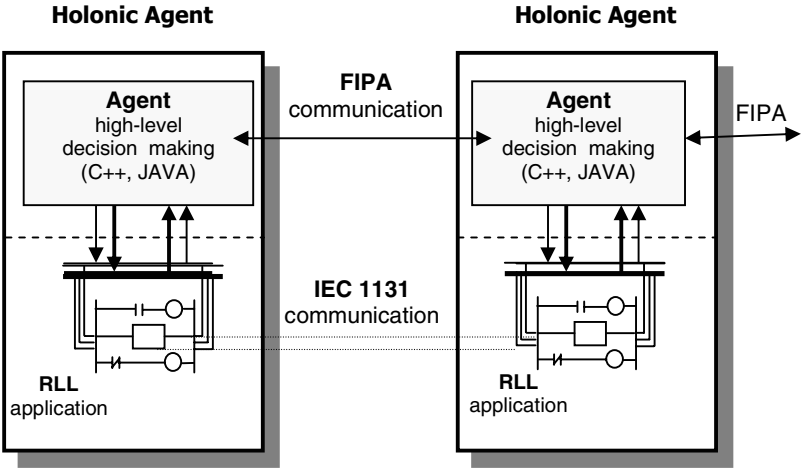


Fig. 2. Model of a PLC-based automation controller with holonic agents using IEC 1131 relay ladder logic for real-time control

In such an agent, equipped with both a lower level real-time component and a higher-level intelligent component, there are three communication channels:

- intra-agent communication between the real-time component and the intelligent component; RA has implemented its agents on its Logix brand of controllers, and the data tables are used for communication between the two components;
- inter-agent communication that is aimed at communication among the intelligent components of multiple agents; RA uses the FIPA standards with its own Job Description Language (JDL) as the content language;
- a direct communication channel among real-time components of the neighboring agents; RA uses the CIP standard for these high speed, deterministic communications.

These agents can therefore widely communicate among themselves, carry out complex negotiations, cooperate, develop manufacturing scenarios, etc., as well as control the manufacturing equipment.

Selling the use of agent-based control systems to customers has proven somewhat problematic. In those discussions RA points out several benefits that its implementation offers over conventional control systems [7]. Moreover, RA discusses with its customers what types of applications can expect to benefit from the use of an agent-based control system. In general, agents are beneficial in control applications where the number of possible configurations of the equipment is impractically large, i.e., too large to accommodate them in traditional programming. This large number can be a result of (i) the nature of the machine or process to be controlled; (ii) redundancy or flexibility built into the design of the solution; or (iii) the large number of combinations of failures that might occur. We offer some examples in the next section, and

then in the following section, discuss some of our observations on existing obstacles to widespread adoption of agent-based control systems.

2 Application Experience

Rockwell Automation has investigated agent-based solutions for a number of applications and has implemented agent-based solutions for two specific industrial applications. The common requirement of these applications is the need for flexibility and reconfiguration. The justification for these projects has been either the increased utilization of manufacturing assets or a more robust system that can continue to operate during major disturbances.

2.1 Rod Steel Production

The first RA industrial agent project involved increasing the machine utilization of a steel rod bar mill. The mill makes steel rods by reheating steel and rolling the steel to size using multiple rolling stands and cooling the steel along a defined temperature profile using multiple cooling boxes as shown in Figure 3. The production process recipes for most of the steel rods require the use of neither all of the rolling stands nor all of the cooling boxes.

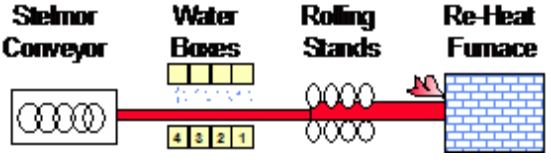


Fig. 3. Bar steel mill process diagram

Hence the system had built-in redundancy and flexibility since it could use any combination of cooling boxes and rolling stands from the subset of working units to produce a given steel rod recipe, as long as the required temperature profile was followed. The aggregate desired behavior of the system was to select and configure a subset of cooling boxes from the working units to satisfy the recipe requirements. This was implemented by enabling each cooling box or unit to assess its own health and bid on its part of the operation. The bids were used with a very accurate simulation of the steel cooling process to enable rebidding until a suitable subset of units and configurations was found as presented in [4].

The agent-based control system did not directly control the bar mill but instead recommended a configuration to the operator. Because of safety concerns and possible damage to equipment the risk was too high to enable direct control by this new technology. Although the agent system preformed very well in all the tests, to release the system for production would require testing all steel recipes with all possible subsets of cooling boxes.

2.2 Navy Chilled Water System

The Office of Naval Research (ONR) was looking for a highly survivable robust control system for the chilled water distribution system, one of the critical ship systems. One of the major requirements was that the chilled water system continues operation even after a major disturbance such as an explosion or a missile strike somewhere on the ship. Several approaches were investigated and a distributed intelligent multi-agent-based system was selected. The main goal was to have a fully distributed system with no single point of failure.

The RA solution was to use the agents consisting of both reasoning and real-time control, distributed among 23 controllers which were physically located near the control hardware. The reasoning part of the agents inside the controllers negotiates the control actions. The intelligent agent control is a good alternative solution to traditional centralized control because it is distributed and the nodes are only loosely coupled. Each intelligent agent provides actions to handle the normal operations but also adds diagnostics and system reconfiguration upon failure. Moreover, the agents can discover what to do when a portion of the system is no longer operational.

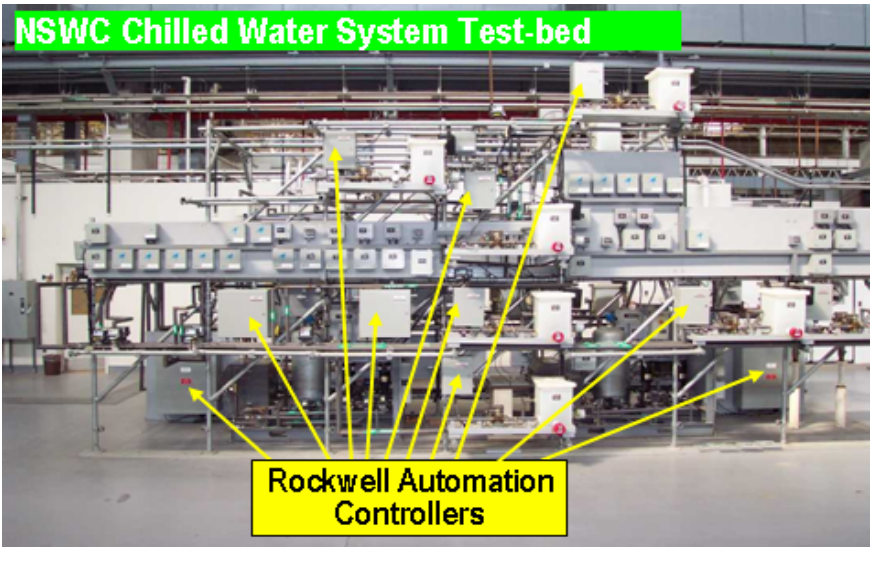


Fig. 4. Office of Naval Research chilled water land-based simulator

In Figure 4, the Navy land-based water cooling system testbed is shown. It includes the plumbing, controls and communications, and electrical components that mimic the real shipboard operations. A typical plan consists of water routes to transport cold water from the coolers to the loads and water routes to move hot water from the loads back to the coolers. The hot and cold water plans, which do not have to transverse the same route, combine to form a complete water route to cool a specific load. The agents evaluate a number of real-time conditions in selecting a water route. Each agent evaluates the physical condition of the equipment under its control to decide its participation in the water routing plan.

After some initial attempts at an agent-based solution, we decided to use an object-oriented methodology similar to "responsibility driven design" as presented in [5]. The resulting system was simpler than any of the previous versions and was easily scalable. To facilitate experimentation and development of the behaviors for the agents we built a "Table Top" physical prototype of the chilled water test system. Though this prototype did provide us information on unknown behaviors such as the directions of the water flow inside the pipes for specific valve configurations, the development of the agent control system was progressing at a slow pace.

We therefore built a Simulink 6.5.1 simulation of the water flow and heat transfer properties of the system to run on Windows XP. The simulation was synchronized with 3 soft controllers (SoftLogix 5800 from Rockwell Automation) running on the same computer. The completed system was simulated in software and then deployed and demonstrated on the ONR land-based simulator shown in Figure 4. RA is continuing its activities on agent-based control systems, but only a relatively few control system customers have demonstrated a real interest. The next section offers some observations regarding some barriers RA sees to widespread adoption.

3 Obstacles to Widespread Adoption

The major obstacle to agent adoption is the risk of a successful project versus the long term payback. The risks for an agent-based system are:

- Can an agent-based system be designed to meet the system requirements?
- Can the aggregate behavior of the agent-based system be guaranteed to meet all the system requirements?
- Can the agent-based system be cost effective compared to a more traditional approach?
- Can an agent-based system be designed by domain experts rather than computer scientists and programmers, and operated and maintained by existing factory personnel?

These questions will be investigated in more detail in the following sections.

3.1 Lack of Skill in "Distributed" Thinking

The agents within a system are each programmed with a set of simple rules so that their aggregate behavior yields the desired system behavior. The root design philosophy is that of object-oriented technology, i.e., programming a distributed system of agents or objects. This "distributed" thinking about control algorithms is difficult to master. Our education system, for example, trains its engineers basically to consider algorithms in a centralized system, i.e., running from start to end on a single computer. There are very few courses in distributed problem solving. RA is working with a number of local and international universities to establish courses in distributed problem solving. We need research into methodologies such as "Responsibility driven design" to enable "evolution" of agent-based systems in a more practical time.

3.2 Determining Emergent Behavior

In designing an agent-based system, a designer typically starts with a set of system requirements. But there is no known formal procedure or algorithm to transform those requirements into the small sets of rules for each of the agents in the distributed system. Nature, on which agents are based, may take millions of years and try all possible combinations, discarding all that fail to survive to evolve the rules for agent behavior. This method, although effective, is not practical in industrial automation. We need research into methodologies such as "Responsibility driven design" to enable "evolution" of agent-based systems in a more practical time. Likewise, given the sets of rules with which each of the agents have been programmed, there is no formal procedure or algorithm to generate the set of resultant system behaviors. Thus there is no guarantee that the system will function as desired. RA has used simulation of the machine or process successfully to view and study the control system's emergent behavior, though building the simulation is at least as much design and implementation effort as building the agent-based control system.

We have observed the emergence of design patterns, similar to design patterns in object oriented design, as we have studied to application of agents to a number of applications. RA, in its agent development environment, employs a template library to remember the design of each class of agent. Each library is application specific, so that each library can remember a set of reusable agent design patterns. We have, for example, a library for chilled water systems that contain design patterns for routing through a network. These "routing" designs might be reused to route electrical power or packages on a set of conveyors. The use of these patterns increases the confidence that an agent-based control system will exhibit the proper emergent behaviors. This use of agent design patterns is an area of continuing study and research.

3.3 Cost of Adoption and Implementation

As mentioned above, for manufacturing equipment to benefit from an agent-based control system, there must be some decisions for the agents to make, i.e., there must be some redundancy or flexibility in the system itself. The capital cost to enable the agents to be useful, such as for a more flexible material handling system or for adding redundant equipment to an existing system, may be prohibitive. This cost may explain why agent technology in the information technology sector, which requires little additional capital expenditure, has been adopted at a much faster rate. The additional hardware costs of an agent-based control system are negligible compared to the capital cost of machinery, though there are some additional software development costs for the implementation and testing of agents. Moreover, to mitigate the risk of adoption, the controls marketplace has traditionally been a conservative one. Most installations must function for many years, and designers want proven technology supported by multiple reliable vendors that have adopted widely used standards.

3.4 Design and Maintainability of Agent-Based Systems

Most industrial automation systems are expected to last in excess of ten years. A major concern is the cost of training and maintenance of any system in a factory. RA chose the IEC 1131 languages for the real-time control aspects of the agents because

that is where the majority of the routine maintenance is required. The great majority of our customers' personnel is familiar with those languages and therefore some of the training costs can be kept low. Expanding an agent-based system may only require more instances of the same agents, but enhancing the behavior of the system requires control engineers skilled in the art of agent design. This emphasizes the need for the educational requirements stated above.

If, in the meantime, intelligent agents are to become useful parts of control system software, suppliers of control systems must augment their programming tools so that customers can easily and in a straightforward manner develop and deploy intelligent agents. The ultimate goal of these tools is to ask customers to provide only application specific knowledge to be stored as the behavior of the intelligent agents; any common behavior must be a permanent component of the hosting platforms. Furthermore, these intelligent agents must integrate easily with traditional IEC 1131 languages. The next section describes some recent improvements to the agent development environment originally described in [6].

4 MAS Design Tools

The Agent Development Environment (ADE) is designed to make agent programming in a distributed system more accessible to application domain experts who are not necessarily computer scientists. It utilizes a declarative style whereby the user describes the agents' behavior and interactions, often choosing items from lists or entering only simple strings and values. Only occasionally does the user need to write procedural code, typically to perform some local action or to extend the system when the desired option is not available in one of ADE's lists.

Up until recently, the agent behavior within ADE could only be viewed and edited structurally. The structure of agent behavior within ADE is as follows.

4.1 Capabilities and Operations

A capability is a set of related behaviors or operations; each message received by an agent refers to exactly one operation. An operation is specified through its signature, i.e., its name plus its collections of inputs and outputs. An agent can support multiple capabilities. For example, for a shipboard chilled water system (CWS), some appropriate capabilities are: ManageShipOperations, SupplyColdWater, and DetermineWaterRoute

In our example system, the capability DetermineWaterRoute has the following operations, written as (outputs) = operationName(inputs):

- (cost, path) = getPath(destination, currentCost, currentPath, nextDestinations, maximumCost)
- () = removeService(serviceName)
- () = canAcceptWaterFrom(currentPath, nextDestinations, waterSource)
- () = allowServiceToOpen(serviceName, loadPriority, supplyPath, returnPath)

4.2 Scripts and Steps

All intelligent agent behavior in this system occurs as a result of receiving a message. A message can come either from another agent, or from the control portion of the same agent. The possible behaviors that an agent can choose to execute upon receipt of a message are encoded in scripts. Upon receipt of a planning message, the agent chooses an appropriate script, based on the

- the requested capability and operation, or
- the values of the inputs of the message, or
- the current state of the agent and environment, or
- any combination of these.

A script consists of a set of steps. Any step within a script is executed as soon as possible, i.e., as soon as both:

- all input values are available for the step, and
- all other steps that have been identified as explicit dependencies have executed.

Each step is either local or global. A local step performs action only on the agent's own data, e.g., reading one's own state, computing some value, or initiating some control action. A global action, on the other hand, causes messages to be sent to some other agent(s) in an attempt to delegate some portion of the agent's activity or to request information. Any agent receiving these messages will, in turn, use their own scripts. Further description can be found in [7]. Results from the use of these scripts are sent back to the requesting agent, who then combines all the responses appropriately to determine whether its global step succeeded or not, plus what values of the step outputs should be used and propagated to the successive steps in its script.

Since in a typical interaction many agent types are involved and many capabilities come into play, it is difficult for the designer to anticipate where in the structure of capabilities-operations-scripts-steps a particular capability or operation is needed. It is much more natural to think about a specific system behavior, and to follow the system's actions from the initial event through the execution of all scripts to the final resolution. To support this functional mode of thinking, recent improvements were added to the ADE, specifically,

- a cross-referencing mechanism whereby all scripts that send a request for a specific capability.operation can be found and listed;
- a cross-referencing mechanism whereby all scripts that can possibly execute as a result of receiving a request for a specific capability.operation can be found and listed; and
- an editor that allows the aforementioned lists of scripts to be edited in place, and for the cross-referencing mechanisms to be reapplied recursively.

Thus the designer can follow the chain of communications among agents forwards or backwards, and edit anywhere he wishes. This functional view matches more easily to the designer's thought processes, and contributes to making ADE easier to use by non-computer scientists.

5 Summary

In this paper we have presented a summary of Rockwell Automation's agent philosophy and work to date on agent-based control systems. Effort within RA continues in the areas of developing more design patterns and application libraries, improving our agent design, simulation, and monitoring tools, using emerging agent related standards, and convincing customers that intelligent agents offer solutions to some of their current and future control problems.

References

1. Koestler, A., *The Ghost in the Machine*, Arkana, London (1967)
2. Auinger F., Strasser T., and Christensen J.H.: Using IEC 61499 Function Blocks (FBs) for Closed Loop Control Applications, International IMS Forum 2004, Villa Erba, Cernobbio, Italy, (2004)
3. IEC (International Electrotechnical Commission), TC65/WG6, 61131-3, 2nd Ed., Programmable Controllers - Programming Languages (2001)
4. Vasko D., Maturana F., Bowles A., and Vandenberg: Autonomous Cooperative Systems Factory Control. PRIMA 2000, Australia (2000)
5. Wirfs-Brock R., Wilkerson B., and Wiener L.,:Designing Object-Oriented Software, Prentice-Hall (1990)
6. Staron R., Maturana F.P., Tichý P., and Šlechta P.: Use of an Agent Type Library for the Design and Implementation of Highly Flexible Control Systems. *8th World Multiconference on Systemics, Cybernetics, and Informatics, SCI2004*, Orlando, FL (2004)
7. Hall K., Staron R., and Vrba P.: Holonic and Agent-based Control. *16th IFAC World Congress,* Prague, Czech Republic (2005)

Fundamental Insights into Holonic Systems Design

Paul Valckenaers and Hendrik Van Brussel

K.U.Leuven, Department of Mechanical Engineering,
Celestijnenlaan 300 B, B-3001 Leuven, Belgium
Paul.Valckenaers@mech.kuleuven.be

Abstract. This paper goes back to the origins of the Holonic Systems concept, a wording coined by Arthur Köstler [1] but actually based on fundamental insights from Nobel Prize winner Herbert Simon [2]. Simon's theme is limited rationality and its implications for the ability to create and sustain sophisticated artifacts in the dynamic and demanding environments that are characteristic for today's society. Holonic and multi-agent systems are amongst the most complex artifacts emanating from deliberate human design and development activities. Therefore, this paper presents these fundamental insights from Simon, augmented with more recent research results on complex adaptive systems [3], and discusses implications for the design of Holonic Multi-Agent Systems. In particular, the development of subsystems (holons) suited for incorporation into larger systems, at some later stage and without knowing these larger systems in much detail, is at the center of the discussions in this paper**.**

1 Introduction

Holonic and multi-agent systems are amongst the most complex artifacts created through deliberate human design and development activities. However, the human designers and developers, involved in these activities, often perceive their results as missed opportunities. This is especially true for the experts that thoroughly master and understand the technologies and subsystems within these complicated and sophisticated systems from the basic elementary building blocks up to the overall system itself.

This paper presents and discusses insights that reveal how this perception of missed opportunities is partly an illusion, which is mainly the contribution from Simon. Conversely, this paper shows how enhanced design principles and development guidelines, based on these insights, open the perspective of creating and maintaining superior systems.

The paper first presents the fundamental insights, revealed by research results on limited rationality [2] and on complex adaptive systems [3]. Next, design principles derived from these insights are presented and illustrated. Finally, some conclusions are given.

2 Limited Rationality and Holonic Systems

Simon's *Watchmaker's Parable* in [2] demonstrates how, in dynamic and demanding environments, the chances of emerging and surviving for systems composed of suit-

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 11 – 22, 2005.

able subsystems are vastly superior to systems composed from basic elements without stable intermediate states or subsystems. In Köstler's words, holonic systems are more likely to emerge and survive in dynamic environments than systems that would ultimately be superior but take too much design effort and development time. The latter systems simply are too expensive and, most importantly, obsolete long before they become operational.

Limited rationality – finite brainpower, bounded information processing and communication capacity – puts a ceiling on the speed at which elements can be integrated to build a system. When these elements are small, more time and effort is required to build a system of a given size. In combination with a dynamic environment, the resulting system is completed too late to be effective or competitive. Systems build from larger building blocks are superior in environments that emphasize adaptation speed over the theoretical possibility of superior ultimate performance.

Implicitly, the above statements assume that integrating elementary building blocks requires similar effort and time to the integration of larger subsystems. In practice, this is largely true but exerts its toll through inferior performance relative to what is theoretically possible. Simon's main goal is to explain why our universe is dominated by systems exhibiting *hierarchical structure in a loose sense* – which Köstler calls holarchies to distinguish from strict hierarchies (typical for rigid manmade artifacts).

Köstler makes a first attempt at characterizing these suitable subsystems, which have to survive the dynamics of the environment better and longer than the overall system (as illustrated in the watchmakers parable). Köstler calls this ability to survive the autonomy of the subsystem or Holon, whereas the contribution of the subsystem to the overall system is called the cooperativeness of the Holon. The autonomy gives the Holon the capacity to cope with changes, uncertainty and disturbances in its environment. In contrast, a subsystem developed in the context of a top-down development activity often is highly constrained in its ability to function outside the specific settings foreseen by its developers. In Köstler's view, such subsystems lack autonomy and therefore are unable to survive in a demanding and dynamic environment. More recent insights in complex adaptive systems [3] reveal however that Köstler failed to discover more insightful aspects of these stable subsystems from which almost every large, complex system in our world is built. These aspects are discussed below.

3 Holonic Systems and Complex Adaptive Systems

In [3], Waldrop describes research activities and insights on complex adaptive systems mainly originating from the well-known Santa Fé Institute. Among the many research results, two properties of complex adaptive systems are relevant for the discussion in this paper: autocatalytic sets and lock-in.

3.1 Autocatalytic Sets

Autocatalysis is more important than autonomy for the emergence and survival of systems, which eventually become subsystems in a larger system. Autonomy and adaptability are only secondary properties of autocatalytic sets, needed to maintain and increase autocatalysis in a dynamic and changing environment.

The concept of an autocatalytic set serves to enhance the probabilities in the standard biologist's theory that life emerged by chance when energy pulses strike a pool filled with organic molecules. Unfortunately for this standard theory, the smallest life forms still are so big that combining its basic molecules by chance is as likely to happen as 'a group of monkeys typing Shakespeare's oeuvre by pure coincidence.' These calculations change however drastically if combinations of molecules are formed into sets that are catalysts for themselves. If energy pulses arrive at a sufficiently high frequency, the autocatalysis implies that the pool rapidly becomes filled with members of such autocatalytic sets (exponential growth until raw material becomes scarce).

The omnipresence of such set members also means that they become the building blocks for larger molecular combinations, amongst which the autocatalytic set members will dominate again. This can be repeated until life forms emerge. If this theory is correct, the dominating life forms should be members of autocatalytic sets themselves. Mice, rabbits, weeds and insects all provide strong empirical evidence supporting the theory. Autonomy, adaptability, manipulating the environment, etc. are secondary properties of the more complex life forms (including humans) that mainly increase the intensity of the autocatalysis in favor of the own set.

To translate the above to the domain of holonic and multi-agent systems, it is necessary to identify the relevant autocatalytic sets for manmade artifacts and software systems in particular. These sets are not situated in artificial worlds inside computer platforms serving to investigate artificial life. The relevant autocatalytic sets comprise both software and humans (i.e. software users and developers). Successful software systems belong to two kinds of autocatalytic sets:

- *The economic set.* Successful software represents economic value to its users and thus mobilizes the economic means for software developers to maintain, adapt and enhance this software.
- *The information feedback set.* Successful software attracts users providing feedback on its shortcomings and merits. This information, in combination with the economic means, allows the developers to maintain, adapt and enhance the software such that it remains competitive.

The above implies that (software) system designers have to account for more than just the technical dimension to be successful. Sufficient users (and their payments) and sufficient diversity in the user community (and its feedback), relative to the competitive pressures, are necessary for emergence and survival.

Most importantly, such successful members of autocatalytic sets are the (only) systems that may become the subsystems in larger more sophisticated systems. Ceteris paribus, software systems with the intrinsic ability to serve more users or a more diverse community of users will have an edge over the competition since their autocatalysis is stronger. Note that software quality and functionality levels are likely to improve significantly through the above types of autocatalysis (personal experience with systems experiencing low levels of autocatalysis has provided the authors with strong empirical evidence supporting this statement).

3.2 Lock-In

The previous section depicts how positive feedback is instrumental in structuring worlds such that larger and more complex systems emerge and survive. This section introduces a negative aspect of such positive feedback: *lock-in into early solutions*.

Systems, in which autocatalysis can occur, may evolve along multiple trajectories where the selection amongst these trajectories strongly depends on which autocatalytic process kicks in the earliest. Since it is an exponential process, the autocatalytic set rapidly exhausts the available 'raw material' effectively eliminating the opportunity for other autocatalytic processes sharing 'material requirements' to start at all. The first process to start generally ends up dominating its world until it fails to survive the dynamics of its environment. At most, a short time window will be available for competing and faster autocatalytic processes to overtake a competitor that started earlier.

In the world of high-tech human-made artifacts, the first product to adequately serve important user requirements typically captures the market and prevents superior solutions from emerging. A well-known example is the VHS standard, being the most inferior technology amongst the competitors at the time. Making the situation worse, lock-in is actively used by commercial organizations to lock out competition. The main mechanism counter-acting lock-in consists of the dynamics of the overall system, making the dominating autocatalytic set obsolete (e.g. DVD overtaking VHS).

Relevant for holonic systems development is the poor level of suitability and adaptability of the available systems (members of autocatalytic sets) from which larger systems have to be developed. Today's systems are developed with specific user requirements in mind, and the world locks into those early solutions. Those early solutions incorporate many design choices that prevent the creation and maintenance of other and larger systems at some later time. Alternative, the later usage for somewhat different purposes of those early systems is much less effective and has lower performance than theoretically possible.

Examples of lock-in in industry are the CNC architecture of machine tools, offering virtually no responsiveness to sensor read-outs and the PLC architecture characterized by poor scale-ability to larger applications. Among others, these early designs consume the available training and support resources blocking out more advanced alternatives. Mainstream computing architectures have comparable lock-in equally reflecting their history (no hard real-time, query-and-answer functionality…). A major difference is that mainstream computing enjoys a more dynamic environment that breaks the lock-in more frequently in comparison to the industrial IT environment. Multimedia en entertainment provide the resources and drive to make mainstream computing escape from obsolete historical design choices whereas industrial IT typically has to work around its legacy. Mainstream computing is likely to conquer industrial IT, which is virtually frozen in those locked-in states.

Overall, lock-in is desirable when it simplifies the world by reducing the options and alternatives that have to be taken into account. Lock-in is undesirable when a dominating system incorporates highly unfortunate design choices, which unfortunately are commonplace since being first (and just good enough in the short run) is more important than being well designed. However, if a developers community is aware of this issue and knows methodologies to handle the lock-in problem better, the

expectation of increased returns from the better designs are likely to prevent really poor designs for achieving autocatalysis to the point that are never developed at all. Hence, the issue addressed in section 4 is how to develop and design systems, without significantly more effort, that are better suited for later usage in other systems and/or larger systems.

4 Designing for the Unforeseen and the Unknown Application

Köstler was overly flattering for the subsystems in Holonic systems. Autonomy is neither omnipresent nor decisive. Autonomy helps a subsystem to adapt and be a competitive candidate to become a cooperative subsystem (holon) within a larger system (holon). Thus, autonomous subsystems probably are more common than others but their autonomy is not the decisive element. Autocatalysis dominates to the point where the larger systems are highly sub-optimal (e.g. incorporate rigid CNC controls) or built from smaller subsystems (e.g. low-level single-axis actuator controls and sensors) decreasing adaptation speed significantly and increasing maintenance efforts.

Systems that become subsystems at some later stage(s) are built with some particular usage in mind. In today's society, virtually every system is developed for incorporation in a particular overall system. The issue addressed below is how to design and develop such systems that they are also suitable subsystems in other systems and better suited for future usage, especially when this extra comes virtually for free (i.e. by replacing arbitrary design choices with purposeful ones).

This section first presents a formal framework, defining problems and their solutions. Next, the reuse of earlier solutions is discussed within this framework. Then, design principles and guidelines are revealed. Finally, sample robust designs for partially unknown requirements are presented and discussed.

4.1 Problems and Solutions

This section formally addresses what constitutes a problem and its solution(s). The purpose of the formal approach is to present ideas more precisely. The formal approach does not produce any calculus on problems and solutions, nor does it claim completeness in a philosophical sense. The formalism mainly serves to avoid ambiguity and to delineate the ideas more sharply than would be possible in natural language.

A problem P is defined as follows:

$$\text{A problem } P \text{ is a constraint on the state space } \mathbf{U} \text{ of the universe } U, \quad \text{defining a set } \mathbf{P} = \{\, \mathbf{u} \in \mathbf{U} \mid \mathbf{u} \text{ satisfies } P \,\} \subseteq \mathbf{U}. \tag{1}$$

When U is the world in which we live, $\mathbf{U}$ is an infinite state space. Every state $\mathbf{u} \in \mathbf{U}$ has a time coordinate $\mathbf{t_u} \in \Re$. By definition, *reachable states at a given time coordinate* are states that either have been or can become the actual state of the universe at this given time coordinate. This universe is subject to the laws of nature (constraints), which limit the number of states that are reachable. These laws of physics imply that there is exactly one reachable state $\mathbf{u}$ for every $\mathbf{t_u} \leq \mathbf{t_{now}}$.

The universe U follows a trajectory through its state space as time progresses. This trajectory is defined for states up to $\mathbf{t_{now}}$. It consists of the states in which the universe has been in the past. The future trajectory is only partially defined. This future trajectory is constrained by physical laws, possibly including stochastic aspects, and is affected by the actions of the human and other entities in this world. These actions affect the choice of the successor states of the current state corresponding to $\mathbf{t_{now}}$. Normally, any significant impact on the trajectory requires sustained action during a substantial amount of time. A problem P is solvable by an agent (human or otherwise) if the agent is able to make this trajectory stay within the given subset **P**.

A solution S to a basic problem P is defined as follows:

Solution S of a problem P is a constraint on the state space **U** of universe U (2)
defining a set $\mathbf{S} = \{ \mathbf{u} \in \mathbf{U} \mid \mathbf{u}$ satisfies $S \}$, where
$\mathbf{S} \subseteq \mathbf{P} \subseteq \mathbf{U}$ and $\forall\, \mathbf{t} \in \Re, \exists\, \mathbf{s} \in \mathbf{S}$: $\mathbf{t}$ = timeCoordinateOf(**s**).

Agents (human or otherwise, intentionally or unintentionally) solve a given problem P when they confine, through their actions, the trajectory of the universe U to the corresponding subset **P**. Therefore, their actions – combined with the laws of the universe – correspond to constraining the state of the universe to a subset **S** of **P**. **S** cannot be empty; it must always have at least one state for every time coordinate. If S fails to comply with this condition, the agents failed to solve the problem.

Typically, **S** and **P** will differ significantly concerning the states with a time coordinate that is smaller than, equal to, or marginally larger than $\mathbf{t_{now}}$. The problem P is only concerned with what is needed/useful/… Therefore, it allows as many states as possible as long as the choice amongst them does not matter – note that this discussion only considers intrinsic aspects and makes abstraction of issues concerning the explicit specification of constraints.

In contrast, the solution S is embedded in the universe, which allows only a single state for every time coordinate in the past (including the present) and imposes severe limitations on what states can be reached in the immediate future from the current state. In other words, **S** will be significantly smaller than **P**, especially concerning states close to the present time and older. Therefore, problem-solving agents have to make choices whenever deadlines approach.

In real life, a problem solving activity consists of a sequence of actions over time. Using the above definition, such sequence of actions corresponds to a sequence of solutions $S_1, S_2, \ldots S_{end}$ that solve P, where $\mathbf{S_{end}} \subset \ldots \subset \mathbf{S_2} \subset \mathbf{S_1} \subset \mathbf{P}$. This reflects that the agents make more and more choices as time progresses in order to solve the problem and comply with the laws of the universe.

An example of the introduction of constraints can be observed in the design of a railway system to solve transportation problems: when the moment of actual usage approaches, the designers have to make more and more choices. For instance, they must select a specific value for the space in between the rails. In fact, the whole problem solving activity can be seen as a sequence of design choices, starting from the

selection of a rail-based system over other possibilities. This introduction of constraints by the solution is the key issue.

4.2 Reuse of Solutions for Unknown Problems

As explained above, complex artifacts (solutions) are constructed using members of autocatalytic sets, which are solutions originally developed to solve other problems. In other words, a solution to an overall problem must be realized by integrating existing candidate sub-solutions (a holonic system is created through aggregation of suitable smaller holons) where these candidates have been developed without knowledge about this overall problem. This section formally addresses this solution reuse process.

Formally, agent **x** solves problem P through solution S_p. Other agents solve problem Q through solution S_q. This results in the state sets $\mathbf{S_p} \subseteq \mathbf{P} \subseteq \mathbf{U}$ and $\mathbf{S_q} \subseteq \mathbf{Q} \subseteq \mathbf{U}$. For instance, agent **x** has constructed the railway system in France to answer the need for transportation. The other agents implemented similar railway systems in the remainder of Europe. These systems are defined as to include the human organization that operates, maintains and adapts/expands the hardware therein. In this example, the overall problem of providing transportation all over Europe – i.e. problem T – is to be solved by integration of the national railway systems – i.e. integration of S_p and S_q into solution S_t. Intrinsically, agents must solve problem T, where $\mathbf{T} \subseteq \mathbf{P}$ and $\mathbf{T} \subseteq \mathbf{Q}$. Practically, agents must integrate the existing sub-solutions into their overall solution; formally: $\mathbf{T} \subseteq \mathbf{S_p} \cap \mathbf{S_q}$. Indeed, society is unable to duplicate the effort of developing their national railway systems to provide an international one. Instead, it must reuse the existing systems, including their ability to adapt, to create the international connections amongst the national systems and obtain a system that transports goods and passengers across the borders in Europe.

The main problem with the creation of such solutions is that the integration fails to deliver good performance. In the example, it is relatively easy to provide international transport at a much-reduced level of service; goods and passengers need to change trains at almost every border. Higher levels of service require significant efforts: locomotives that support multiple power supply systems, railway equipment supporting variable width of the railway tracks... The sub-solutions, which were developed independently, have made mutually incompatible design decisions and these decisions have accumulated significant inertia (i.e. it has become costly to undo these decisions). Formally and with an ambitious overall problem T_A, "$\exists$ t $\in \Re$, $\forall$ s $\in S_p \cap S_q \cap T_A$: t $\neq$ timeCoordinateOf(s)" expresses that, in practice, reuse of sub-solutions cannot be reconciled with an ambitious overall problem T_A that requires a high-performance solution, as if the sub-solutions were redeveloped with problem T_A in mind.

In practice, a solution for T_A cannot readily reuse the available (partial) solutions offered without undoing a lot of design choices. Typically, society only receives a reduced level of service (i.e. solutions to easier problems), and will only gradually outgrow the old designs when technology progresses sufficiently to introduce a new solution from scratch (e.g. high-speed trains). The key issue is the introduction of

constraints that are absent in the corresponding problem and that may cause future integration problems. More precisely, it is the accumulated inertia – i.e. the effort necessary to undo such harmful design decisions – that constitutes the problem.

4.3 Designing for Unknown Problems

From the above, it becomes clear that the problem-solving agents must design a solution S_p capable of surviving in an uncertain environment concerning its future. Formally, such uncertain environment corresponds to $\wp$, a set of subsets of the state space $\mathbf{U}$. The actual future will offer an intersection of members of $\wp$ as the space that is available for S_p to contribute its part to the overall solution. Some members of $\wp$ correspond to the constraints imposed by the future problems for which solution S_p may be part of the overall solution. Alternatively, members of $\wp$ correspond to the constraints imposed by other candidate sub-solutions that may contribute to solving the bigger problem at hand. A designer of solution S_p does not know which members of $\wp$ will be present, and must try to avoid conflicts with any constraints that might be presented by members of $\wp$.

In this context, design decisions can introduce two types of constraints: stable and unstable. Stable constraints will be present in all conceivable future situations within the scope of S_p. For instance, an agent may assume that power supply will be 240V/50Hz or 130V/60Hz when designing an electrical appliance. Formally, no member of $\wp$ will be in conflict with the stable constraint. In contrast, unstable constraints represent conflicts with some members of $\wp$. Design decisions that introduce unstable constraints reduce the future capability of the solution. For instance, the designer of a rocket inertial navigation program may choose to limit the range of the acceleration supported to the limits of the current rocket, causing the crash of the first rocket of the next generation when the constraint remains undetected.

Based on the above, design principles P1 and P2 emerge. Note that these principles apply when designing lasting artifacts, like infrastructures, and not for the short-time solutions for the immediate future.

P1: Problem solvers must avoid introducing potentially harmful constraints. This guideline is twofold. First, it advises against the introduction of (any) constraints. As seen in section 4.1, this cannot be avoided for the immediate future. However, concerning problem solving in a slightly more distant future, systems implementing low and late commitment strategies avoid the introduction of constraints. This is a relatively well-known fact, and it is widely adhered to. Unfortunately, cost and complexity of such solutions seriously restrict the applicability of this design principle. For instance, building locomotives and railway wagons for variable-width tracks is prohibitively expensive/complex. Software related to the railway system has less difficulty to avoid commitment to a specific track width; it simply becomes a system parameter to which the remainder of the software must consistently adapt.

Second, the guideline encourages introducing constraints unlikely to cause future conflicts; these are called *stable constraints or decisions* in this manuscript. In short, the introduction of stable constraints simply reflects the fact that the constraint is

already present in the environment; a stable design decision preserves and reflects the scope of the problem domain. In a way, these stable constraints can be seen as an extension of the laws of nature for the application domain at hand.

P2: Problem solvers must avoid/reduce the inertia build-up for potentially harmful constraints. Again, this guideline is twofold. First, designers are encouraged to implement low-inertia versions of their solutions first and only to implement the real system when, hopefully, most conflicts among design choices have been undone in this low-inertia version of the design. This is a well-known principle and specific technologies have been developed to support this: CAD, CAE, … Indeed, it is significantly less costly to undo a design choice on a CAD drawing of a bridge than to do the same modification on the bridge itself at a later stage. Likewise, virtual reality technology is used on a regular base to discover and undo conflicting constraints in factory designs. Similarly, in the research on multi-agent manufacturing control, control software prototypes will be connected to emulations of the underlying manufacturing system first, allowing adaptation of both the control system and the underlying system in the virtual world before committing in the real world.

Second, designers must avoid reinforcing unstable constraints introduced by earlier unstable design decisions. Designers cannot avoid introducing unstable constraints, e.g. selecting a width for the railway track, when the time for implementing the solution approaches. However, the fact that this width needs to be fixed before the first locomotive can be build does not imply that this width needs to be fixed in other situations, e.g. being hard-coded in a piece of embedded software, where the cost of keeping your options open is negligible. Whereas stable constraints can be reinforced without problems – the environment or general scope of the problem domain already imposes such constraints – an earlier unstable constraint gains inertia every time another design decision introduces it again. For instance, it is a known problem that highly automated production facilities are hard to adapt because their control systems (= software) reinforce the existing way of operating and require significant maintenance for every change in the system. If the software for a control system expects a tree-shaped bill-of-materials, the introduction of disassembly operations will be problematic, especially when the software has a functional decomposition design. Ironically, the technology that is seen to be extremely flexible causes the rigidity (because of its poor design).

In summary, the novel principles for the designers are: (1) designers must prefer stable design decisions and (2) earlier unstable design decisions are no justification for later decisions imposing the same constraint(s). The first design principle avoids the introduction of new constraints. The second avoids the build-up of inertia for unstable constraints that were introduced earlier; every unstable design decision must be justifiable by itself. Earlier unstable design decisions only affect which later unstable design decisions are taken (i.e. compatible with these earlier ones), not whether a later unstable decision will be taken at all. Next to these two items, some better-known principles – low and late commitment, autonomous systems, and low inertia implementations – remain valid. Note that low commitment includes the ability to fall back on interim designs at stages before unstable constraints are introduced.

4.4 Sample Robust Designs for Unknown Problems

Although the scientific and engineering communities are largely unaware of this, our society has successfully created and developed subsystems that are well prepared for the unknown, or perhaps more precisely, are usable in a wide range of applications. This section discusses examples with varying degrees of success (or compliance with the above design principles), the key being not to rely on the wrong assumptions for a subsystem to be a valid (part of a) solution.

Physics
Physical science is an extreme example of compliance with the above design rules. At a first glance, it does not make any choices at all – apart from representation/language/symbol aspects. Looking more closely, it is possible to distinguish an applicability range. For instance, Newton's classical mechanics become invalid when velocities approach the speed of light or when dimensions become extremely small.

Science uses the ultimate source of stable constraints: the world itself. A collection of artifacts – in this case scientific theories – reflecting parts or aspects of the real world will not conflict within the scope in which they are valid. If conflicts occur, these conflicts are caused by flaws in the artifact, not by unstable design choices. Indeed, it suffice to look at the world – by means of suitable experiments – to know which theory is correct (or to discover that all are flawed).

Navigation and Transportation
Consider the following types of artifacts: route descriptions, traffic regulations and maps. Maps clearly comply with the design principles, assuming they reflect relatively long-lived aspects of some part of the world. Note that they benefit from the same source of stable constraints as scientific theories – the real world. When two maps contradict each other, at least one map contains a mistake and verifying against reality suffices to solve the conflict. Notice how maps covering overlapping sections of the world and representing overlapping aspects (e.g. tourist maps, traffic maps, military maps) can be used inside a single larger system/application. Maps augmented by an organization (including humans) to keep these maps up-to-date at adequate frequencies represent an even better solution in line with the design principles.

Route descriptions are less robust artifacts in two ways. First, route descriptions are exposed to highly unstable user requirements (origin, destination, route attributes), whose number of possible combinations is practically unlimited. Second, route descriptions are fragile with respect to sensing accuracy, disturbances and changes. When its user confuses an exit of a parking lot with a side street, route descriptions often provide a poor level of service. An even worse problem occurs when the route, indicated in the description, is temporarily blocked by road works. Traffic regulations are possibly even worse. Resolving arbitrary choices is a main reason for their existence, and it should be no surprise that combining two regulations, which came independently into existence, more or less results in undoing half of the design choices.

Software Design
Today, two major paradigms for software development exist in practice. Historically, top-down functional decomposition preceded object-oriented design. Functional design starts from functional user requirements and decomposes these in a top-down fashion. User requirements are notoriously unstable. As a consequence, subsystems

developed by functional approaches are likely to incorporate a multitude of unstable design choices. They are the analogue of the route descriptions above. Remark that they do not only share the presence of many unstable design choices (a negative aspect) but also require roughly the least effort possible to answer their original user requirements (a positive aspect).

In contrast, object-oriented design includes the elaboration of an essential or conceptual model of the area of interest [4][5]. In an object-oriented design of a navigation application, this essential model would comprise a map of the area in which the navigation occurs. In personnel administration applications, the essential model would reflect the possible life cycles of personnel. Again, the real world serves as a source of stable constraints. As with maps in navigation, the elaboration and implementation of such essential models often requires too much effort for single-use applications. However, the object-oriented approach rapidly becomes superior when user requirements are partially uncertain and the application needs to survive beyond the immediate future.

Multi-agent Systems

Often, the analytical minds of DAI researchers have been attracted to 'consolidated user requirements addressed by a MAS solution'. As a consequence, much MAS research starts from the top-down functional decomposition paradigm and fails to develop suitable subsystems from which larger systems can be composed. Symptomatic for this phenomenon are researchers calling out for solutions that are as small as possible, focusing on decision-making, automatic generation of MAS, methodologies to keep development efforts small and short.

In contrast, a number of researchers are investigating agent systems in which agents reflect part of the relevant reality: "agents must be things" [6]. This is especially true for the MAS designs that are bio-inspired [7] and/or searching for emergent functionality and self-organizing applications. When selecting these parts of the world that are to be reflected by the agents in a MAS, it is important to safeguard the possibility for strong autocatalysis (i.e. user community must be able to support and maintain the software artifact). The PROSA architecture is an example of a design aimed at maximizing the user communities of the agents therein [8].

5 Conclusion

This paper discusses the fundamental nature of holonic systems. Limited rationality implies that sophisticated systems in a dynamic and demanding environment mainly consist of large subsystems [2]. Candidate systems, available to become such subsystems, belong to two types of autocatalytic sets [3], and therefore are predetermined to a significant extent. This limitation often results in poor performance for large systems when judged against the performance that is theoretically achievable with custom-made subsystems. The latter would be obsolete by the time they are available or fail to achieve adequate autocatalysis because the existing systems have drained the world of the suitable resources. This paper analyses the root cause of this sub-optimal performance and derives design principles and guidelines to remedy this situation.

Acknowledgements

This paper presents work funded by the Research Council of the K.U.Leuven – Concerted Research Action on Agents for Coordination and Control.

References

1. Köstler, A.: The Ghost in the Machine. ARKANA S. (1990).
2. Simon, H. A.: The Sciences of the Artificial. MIT Press, Cambridge Mass. (1990).
3. Waldrop, M.: Complexity, the Emerging Science at the Edge of Order and Chaos. VIKING, Penguin group, London.(1992).
4. Cook S., Daniels J.: Designing Object Systems. Prentice Hall, London. (1994).
5. Jackson, M.: Software requirements and specifications. Addison-Wesley, (1995).
6. Parunak, H.V.D.: "Go to the Ant: Engineering Principles from Natural Agent Systems", Annals of Operations Research, 75:69-101, (1997).
7. Grassé, P.P.: La theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs, Insectes Sociaux 6, (1959).
8. Van Brussel, H., J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters: Reference architecture for holonic manufacturing systems: PROSA. Computers In Industry 37, 255-274, (1998).

A 3D Visualization and Simulation Framework for Intelligent Physical Agents

Jose L. Martinez Lastra[1], Enrique Lopez Torres[1], and Armando W. Colombo[2]

[1] Tampere University of Technology, Institute of Production Engineering, P.O. Box 589, FI-33101 Tampere, Finland
{jose.lastra, enrique.lopez}@tut.fi
[2] Schneider Electric GmbH, Steinheimer Str. 117, 63500 Seligenstadt, Germany
armando.colombo@de.schneider-electric.com

Abstract. MAS (Multi-Agent Systems) and HMS (Holonic Manufacturing Systems) are enabling the vision of the Plug & Play Factory and paving the way for future autonomous production systems. Previous research activities and pilot implementations reported the FIPA (Foundation for Intelligent Physical Agents) set of standards as a facilitator for the interoperation of heterogeneous agents in manufacturing control and planning. One of the current challenges for the design and implementation of intelligent agents is the simulation and visualization of the agent societies. This issue is significant as soon as the software agent is embedded into a mechatronic device or machine resulting in a physical intelligent agent with 3D-mechanical restrictions. This paper provides a review of the current software tools for the simulation and visualization of MAS in industrial applications and presents a software suite for the three-dimensional creation, simulation and visualization of agent societies. The documented research describes the methodology for the 3D representation of individual agents, the related identified objects present in the interaction protocols, and the assembly features and clustering algorithms. Finally, the paper describes how to capitalize the 3D information for controlling and planning of industrial assembly operations.

1 Introduction

Current global trends that significantly affect manufacturing practices are those of shorter product life cycles and mass customization. In order to respond to these market requirements, the factory floor needs to be set-up for new products in very short times. Whether planning new plants or retrofitting existing ones, highly-flexible and highly-reconfigurable working environments are needed [1]. However, flexibility and reconfigurability of the working environment often conflict with the requirement for high productivity. The solution is to migrate from conventional factory floor control strategies to flexible and collaborative automation systems. The hierarchical management and control philosophy needs to be broken down into intelligent, collaborative and autonomous production units [2], these units are intelligent physical agents.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 23 – 38, 2005.

During the last decade, the application of MAS and HMS in factory automation has targeted different control activities such as control of material flow or control of flexible manufacturing cells and systems including the real-time scheduling. Extending the traditional concept of software agents to autonomous physical units the proposal of an intelligent mechatronic device called actor was developed by the Institute of Production Engineering at Tampere University of Technology. This concept has been applied to a specific manufacturing domain, i.e. assembly which results in Actor-based Assembly Systems called ABAS® [3]. ABAS are reconfigurable systems built by autonomous mechatronic devices that deploy auction- and negotiation-based multi-agent control in order to collaborate towards a common goal, the accomplishment of assembly tasks. These assembly tasks are complex functions generated as a composition of simpler activities called assembly operations which are the individual goals of the actors.

In order to provide an environment to emulate actors, societies and their behavior, to assist in the design of a new system this paper presents a framework for visual modeling and simulation of intelligent physical agents, this environment consists of two software tools. The first tool addresses both the modeling and the emulation of physical intelligent agents, called actors, either as stand-alone components or as participants in complex societies. The second tool serves as a runtime platform where actor societies can be deployed and visualized in a 3D environment, and which performs executive control of assembly processes. The platform can serve as a runtime environment for physical and/or emulated actors indistinguishably.

The rest of this paper is structured as follows. Section 2 reviews prior work in the field of MAS and HMS from a visualization viewpoint. Section 3 discusses theory of Actor societies and formation of Actor clusters. Section 4 introduces the proposed 3D framework by presenting the: Configuration and Visualization Tools; and the common elements. Finally, Section 5 outlines the conclusions for this work.

2 Previous Work in Simulation of MAS and HMS

This section reviews previous work in the field of MAS and HMS within the manufacturing domain and outlines the most promising reported simulation and visualization tools such as MAST (Manufacturing Agent Simulation Tool) [4].

Multi-Agent Systems are applied to manufacturing control at several levels. When agents are integrated with the physical manufacturing hardware, referred to as *holons* or *holonic* agents, they enable distributed and collaborative real-time control at the factory floor. Other types of intelligent agents, usually implemented as stand-alone software units, are also used at the production management level. Moreover, agent-technology is also applied to the Virtual Enterprise, an abstraction used to represent conglomerate of enterprises that complement each other with a particular business objective [5].

The implementation of agent technology at the paint shop of the General Motors Corp. was a milestone in the use of artificial intelligence for the control of manufacturing activities. This implementation was significant in that it defines agents within the architecture as not only traditional software elements but also as individual physical devices, such as humidifiers, burners, chillers, and steam [6].

Even though agent-based control is a new approach, the potential advantages that can be achieved make it possible to implement other applications such as those at Daewoo Motors in which the task, resource and service agents interact in a market-driven approach, building communities via hierarchical aggregation [7]. The field of planning and scheduling has been very receptive to this new technology. However, implementations in the field of real-time control are also common in the literature. Another example of an agent-oriented application is AMROSE in the application domain of shipbuilding, this application is especially interesting since it not only belongs to the field of robotic control but also builds the robot by assigning an agent to each of the links, with each agent deriving its positioning goal from the next agent closer to the end effector [8]. An exhaustive review of different industrial and practical implementations has been compiled in [9]. According to [10] three different approaches have been adopted in agent-based control: 1) Auction- and Negotiation-based; 2) Artificial Markets; and 3) Stigmergy and Ant Colony Coordination. These approaches differ in that the first two agents explicitly interact and negotiate with each other, while the latter one belongs to the category of agents that indirectly interact and coordinate by changing their environment.

The HMS consortium delivered an interactive, virtual simulation of a Holonic material flow simulation tool. The tool targets an automated workshop production, where the material flow is carried out by Automated Guided Vehicles (AGVs). As result of the research effort under the Intelligent Manufacturing Systems (IMS) framework Rockwell Automation in cooperation with different partners has designed and developed MAST a graphical visualization tool for multiagent systems reported in different forums. The main target is the materials handling domain and it is built on the JADE standard FIPA platform. In MAST, the user is provided with the agents for basic material handling components as for instance manufacturing-cell, conveyor belt, diverter and AGVs as Figure 1 shows. The agents cooperate together via message sending using common knowledge ontology developed for material handling domain.

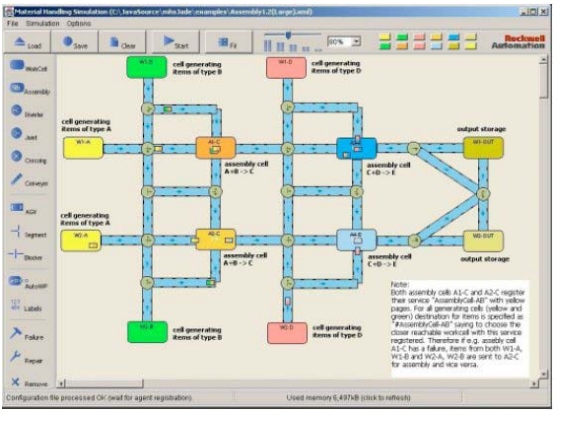


Fig. 1. A reported version of MAST (Manufacturing Agent Simulation Tool) by Rockwell Automation [4]

MAST represents the state of the art in graphical simulation tools for modeling and simulation of multiagent systems in manufacturing control, however and due to the fact that only material handlings systems are targeted the tool does not cover complex application from a 3D geometric viewpoint such as the robotic manipulation.

3 Three-Dimensional Models and Visualization Mechanisms

In order to deploy ABAS systems, a set of architectural elements have been defined. These are the Actor, Register, Recruiter, and Actor Cluster. The functionality of these is reviewed in the following paragraphs.

Assembly actors are mechatronic entities built on a well-defined set of resources for accomplishing typically one assembly operation. In order to participate in an ABAS society, actors must report themselves to the Register, described below. Actors then perform assembly operations as requested by Recruiters, also described below. Actors also contemplate the collaboration with other actors, and have appropriate behavior to cope with that kind of situation.

The Register is a software entity that resides in the ABAS platform or may be distributed along the different actors of the ABAS society. The overall goal is to regulate the admission of actors into the society and to keep a record containing the different members accommodated on the given society and their addresses in a "white-page-list" manner, in addition the register provides to ABAS with a "yellow-page-list" type of services for each registered actor. This functionality is similar to the one provided by the Directory Facilitator component defined by FIPA.

Recruiters are software entities that seek to enroll and also secure the accomplishment of assembly tasks by recruiting assembly actors with the necessary skills for executing the requested assembly operations. Three recruiters work sequentially for the completion of a composing task. The first recruiter deals with the operations required for the part that will be composed together with an assembly. The second recruiter deals with the operations that deal with the assembly where the part will be composed. The third recruiter is in charge of the required operations once the part and assembly become a single entity.

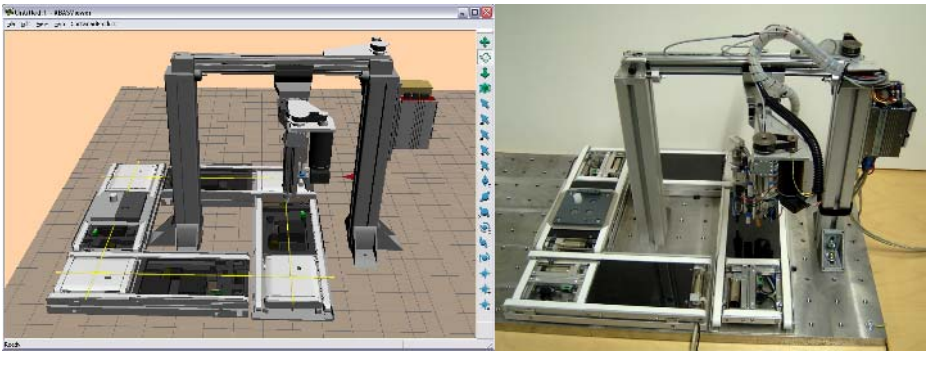


Fig. 2. A 3D view of a system built by intelligent physical agents (ABAS Viewer) and the correspondent physical implementation

ABAS recognizes, via the figure of the recruiter, the actors capable of the solving the required assembly needs by grouping the individuals into a set of actors called cluster. Thus, clusters are software entities that keep record of the potential groups having the necessary skills for solving a given problem. There may be more than one cluster for a given job as it was pointed out previously. The life cycle of the cluster ends once the requested job has been performed [11]

3.1 Individual Physical Agents: Actors

Actors can be considered intelligent physical agent, being intelligent physical agents defined as mechatronic devices augmented with agent behavior. Each actor is assigned, as functional objective, one of the assembly operations of figure 4. This figure also illustrates the assembly operations needed to perform complex assembly tasks. These tasks can therefore be accomplished by forming Actor Clusters, where each composing actor provides one of the required operations.

In order to achieve the desired individual functionality and the necessary skills for combining this functionality with the ones exhibited by others, each actor has as set of resources. These resources are computation, communication, actuating, and sensing resources and are used for providing the necessary internal services. Thus, the actor will be able to model the environment where is placed and its current status within it by making use of the sensing resources. The actor will also control its environment by executing different control algorithms that have been encapsulated into software modules, these software components are executed using the actor computational resources, and as consequence of those control laws the actor will use its actuating resources in order to change its environment. At any time, the actor must ensure the necessary communication links in order to interact within the society that is a member, this requirement guarantee by the external communication resources that the actor has.

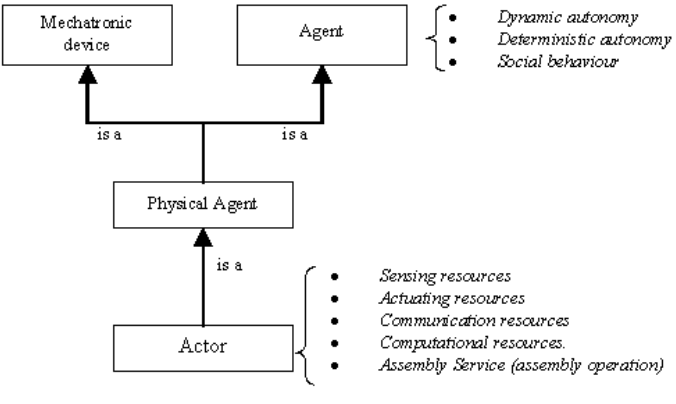


Fig. 3. UML- Actor, Agent, Object and Mechatronic relationship

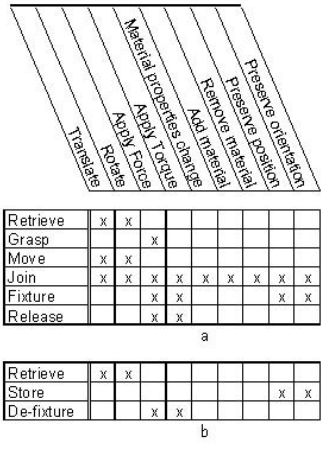


Fig. 4. Operations generally present in each assembly task related to a) Parts and b) Assemblies/products

3.2 Agent Societies: ABAS Systems

One of the key features of ABAS systems is that when complex assembly processes is introduced to the system, a set of Recruiter entities have the responsibility of identifying the basic assembly operations that compose the process. The recruiters enroll actors existing in the ABAS society that capable of providing those basic operations, and thus the complex process is fulfilled. When a group of actors collaborates in this way to perform a complex process, they are grouped into a temporary structure called Cluster. The problem addressed in the following section is that of cluster construction.

In order to construct clusters, recruiters need to distinguish the relations between actors in order to know how can they work together to accomplish a certain output. In other words, recruiters must understand how an actor can affect its surrounding environment (other actors) in order to accomplish the required goal. For recognizing the relationships between actors, the use of assembly features is proposed. Features have been originally used in modeling and planning for manufacturing of parts. Such features combine geometric and functional information. This work follows this approach and proposes the use of assembly features as the basis for modeling actor clusters as an assembly of actors. Combining actors in order to form societies aligns well with the definition of assembly actions, putting together components to form a more meaningful entity. Analogies, therefore, can be constructed by viewing actors as parts or societies, and assemblies of those parts. The proposed assembly features of actors are: Location, Physical dimension, Working dimension and Actor Interfaces. These, together with the service type which defines the basic assembly operation that the actor is able to perform, form the common attributes of any actor. Table 1 illustrates those attributes and their format, the last four elements of the table can be considered assembly features since they enclose functional and geometry information.

Table 1. Actor Attributes, including Assembly Features

Assembly Feature	Format	Abbreviations
Service Type	Assembly Operation	ST
Location	$P_l\ (x_l, y_l, z_l)$ $\theta_1, \theta_2, \theta_3$	LC
Physical Dimension	$P_{1p}\ (a_{1p}, b_{1p}, c_{1p})$ $P_{2p}\ (a_{2p}, b_{2p}, c_{2p})$	PD
Working Dimension	$P_{1w}\ (a_{1w}, b_{1w}, c_{1w})$ $P_{2w}\ (a_{2w}, b_{2w}, c_{2w})$	WD
Actor Interface	$P_r\ (x_r, y_r, z_r)$ $\{R\}\ \theta_1, \theta_2, \theta_3$	AIFC

The presented features are used to create relationships between actors. Those relationships define the effect between actors and how they can be requested to perform certain assembly task. The form in which those features can be known by different actors is by using the FIPA query interaction protocol

Location. The location in space of an actor involves 3 coordinates for the position and 3 angles for the orientation. ABAS systems, as in any robotics manipulation system, by definition imply that parts (products) and tools (actors) will be moved in space. This naturally leads to the need for representing position and orientation of those entities, in other words their location. The location is necessary information for recruiters given that the rest of the features (WD, PD and AIFC) are related to the actor coordinate system, with origin in the actor location. In most cases it is needed to refer (transform) those features to the world coordinate system (WCS) so that they can be compared with the features of other actors. The information provided by the location represents the actor coordinate system related to the world coordinate system (WCS), and can be used to create homogeneous transformation matrices that transform those points related to the actor coordinate system to the WCS. The homogeneous matrices are useful when the calculations are performed by a computer, since it is possible to express transformations of vectors with matrix products, instead of combination of additions and multiplication of matrices.

Physical Dimension. The physical dimension is represented by 6 scalars that provide the coordinates for two points. These two points define the diagonal of an orthogonal hexahedron containing the actor's body. The physical dimension (PD) can be composed of more than one hexahedron (or bounding volume).

Working Dimension. Using the same format as the PD, the working dimension (WD) represents the potential dimension of actuation under the service type operation that the actor can perform.

Actor Interface. The Actor Interface (AIFC) provides information in order to determine the current relative location between different actors. This information is calculated and defined when an actor is designed and represents reference points actors, e.g.: home positions in the servo actuators flag-points at the transportation units, such as loading and unloading points, waiting points, etc.

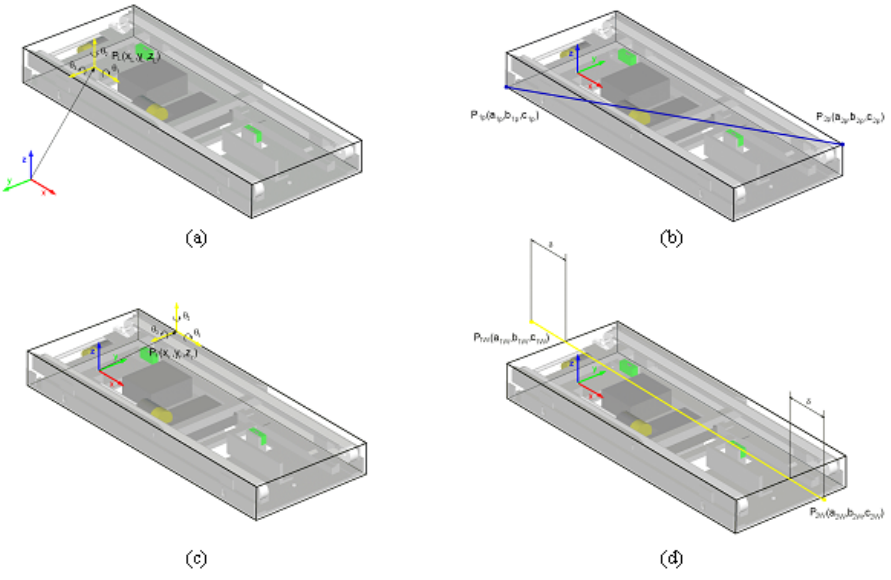


Fig. 5. Example of the Actor Features identified in an actor prototype. The Service Type for this prototype is the translation of assemblies, parts and products.

The AIFC goal is to reduce the location uncertainties when motion between actors and products is started. In this research the AIFC is represented as a 3D coordinate frame, this approach is justified due to:

- The possibility to use it as a PD vertex mating, due to the PD are represented by hexahedrons (bounding boxes) it is possible to fit one of the eight vertexes of an hexahedron with the AIFC of another actor. This scenario is illustrated by Figure 6.
- It can be used as frame mating between products and actors where the coordinate system of parts matches with the AIFC. In this case the AIFC is used as a frame as it is illustrated by Figure 7.

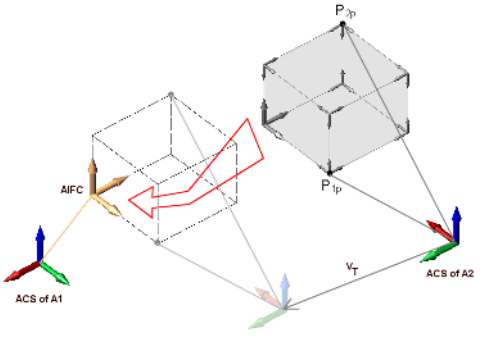


Fig. 6. Matting PD vertex of actor A2 with the AIFC of actor A1

The Figure 5 shows all the identified objects of an actor prototype including: a) actor location; b) actor dimension; c) actor interface; and d) working dimension.

3.3 Actor Contact Features

A formal definition of contacts is introduced in this section. Figure 8 shows the possible contact types (CT) between two actors. Contacts can be defined as assembly features since they represent functional and geometric information. Contacts not only have information about what actor dimensions (PD and WD) are in contact but also the geometry formed by that contact. That contact geometry or contact dimension (CD) is also represented with hexahedrons. By knowing the contact type (CT) it is possible to know the effect of certain actor(s) over other actor(s). Table 2 presents the functional information of actor contact features.

When two actors have more than one CT and there is a physical contact (PD-PD) type between them, it is necessary to discern if they are either rigidly attached or not. Two actors can be in physical contact because one of them is sliding over the other one, i.e. a container actor that is sliding along a transporter actor (non-rigidly attached). On the other hand, it is necessary to consider the situation where an actor body is attached to another actor body without any WD between them (rigidly attached). Therefore, it is necessary to prioritize the contact detection between actors since it does not mater in some cases that there is a PD-PD contact since another CT defines the real behavior between actors. Table 2 shows the priority level that is defined for each CT.

The functional information enclosed by actor contact features presented in Table 2 is needed by recruiters for two reasons:

- The recruiter will know the effects of certain actor(s) over other actor(s), and therefore it can calculate the necessary requests to them. By using the CD geometry it is possible to calculate the distances related to each actor.
- It is possible to use the CD in order to calculate WD amplifications (in case that they exist), and therefore to know it the cluster is capable to perform certain assembly task.

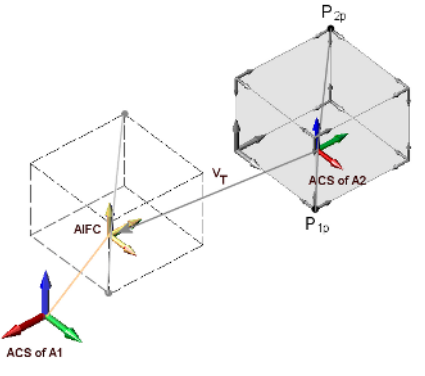


Fig. 7. Matting frame of actor A2 with AIFC of actor A1

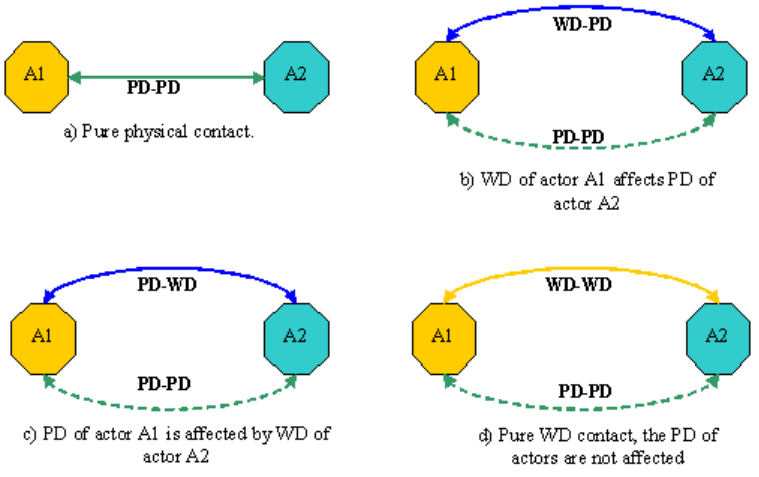


Fig. 8. Representation of all possible contacts between two actors A1 and A2

The priority helps the recruiter to choose one interpretation out of the detected contact types between certain actors.

Contact Detection. The method used for contact detection is similar to the collision detection methods used in computer simulation field [12, 13]. When having objects of different shapes moving in a simulation environment, it is sometimes necessary to determine not only if either those objects are colliding or not but also the resulting collision dimension. Figure 9 shows the class diagram for the specification of the collision dimension.

The contact detection algorithms depend on the representation of the object's bodies in the computer environment. For two dimensional shapes, for example, a collection of points connected each other with lines can be defined e.g. hexagons are formed by six points. However, for three dimensional shapes the representation of those bodies is more complicated, where a simple collection of points is not enough. One way to achieve the representation is to combine basic 3D shapes as spheres, boxes, cylinders, cones, etc. Putting these together would form a more complex 3D

Table 2. Contact assembly features

Contact Type	Functionality	Detection Priority
PD-PD	A relocation in actor A1 may relocate with the same magnitude the location of actor A2	Low (Rigid attachment)
WD-PD	The WD of actor A1 can affect the PD of actor A2	Normal (Non-rigid attachment)
PD-WD	The body of actor A1 can be affected by the assembly operation of actor A2	
WD-WD	Actor A1 and actor A2 are able to cooperate without affecting their PDs	High

shape. However, this approach requires storing different types of data associated with the specific basic shape, e.g. diameter for spheres, height, width and deep for boxes; this approach becomes challenging when representing complex shapes. A widely used technique to represent complex three dimensional shapes in computer environment is the technique based in triangles. Analogue to storing points for two dimensional shapes, regarding three dimensional shapes it is possible to store triangles; each of these triangles forms a plane, and by combining all those triangles together they can form complex shapes. By increasing the number of triangles used to form complex shapes it is possible to create more detailed shapes that describe more accurately the real object body. This approach though requires more computational power for operations such as translation, rotation, screen rendering of those objects as well as for collisions detection between them. This increase in computational power also includes the case of calculating the resulting collision area or contact of those kinds of bodies. The idea of finding a way to represent in the most optimized way the body of an object in a virtual environment lead to a kind of structure that can be used not only for the simplification of algorithms but also to use of a generic structure. Referring to the class diagram of collision dimension, it is possible to see that a collision dimension can be specified as a body, surface, curve or point. A body is composed by surfaces, those surfaces are composed by curves and curves are composed by points. Going to a more specialized level, a solid is a kind of polyhedron which is composed of faces, those faces are composed of edges, and finally those edges are composed of points. In the same way as with the use of evolving triangles introduced before, those faces are the surfaces defined by each triangle; the edges are those lines that connect the triangle points. Finally in the same class diagram it is presented the hexahedron as a special case of a polyhedron in which the hexahedron can be defined with two 3D points that represent the opposite corners of it. Therefore the use of hexahedrons is useful when representing planes, lines and points as special cases, for example a plane can be seen as an hexahedron with volume zero but with certain area, or a line as an hexahedron without neither volume nor area, and finally a point in which the hexahedron's points are the same. This is useful when defining a generic structure for representing different shapes in which the principle of substitutability can be used, since the hexahedron structure would represent a body, surface, line or point. Therefore the representation of the PD and WD as hexahedrons facilitates the design of algorithms for collision detention and it reduces the computational power required for executing such algorithms. Moreover, the resulting collision body (or contact dimension - CD) as a result of collisions between WD and PD in any combination would result in another hexahedron (or any of its special cases). The disadvantage in using bounding volume would be the inaccuracy.

4 Implementation

Two software tools that facilitate the design and deployment of ABAS systems have been developed at Tampere University of Technology. These tools are, respectively, the ABAS WorkBench and the ABAS Viewer. A reusable software component, the Actor Blue Print, has also been implemented in order to facilitate the development of Actors. These tools incorporate the theory and concepts developed in Sect. 3 for

actors and actor societies. These tools are implemented using Visual C++ and deploy an in-house implementation of FIPA Interaction Protocols. The theory for the cluster formation is not described in this paper but can be found in [3] and [9].

4.1 ABAS WorkBench

ABAS WorkBench is a tool used for designing and emulating ABAS systems, from the atomic design (actor) to the system level design (actor society). The goal of this software is to provide to designers with the ability to produce actor prototypes and experiment with them before the real implementation.

As shown in Figure 10, the ABAS designer is able to create actor prototypes, which can be tested later in an emulated society; if an actor needs to be refined it is possible to edit it until the desired behaviour is achieved. When an actor prototype is modeled the resulting information is stored in flat files (text files) that can also be used in the implementation of real actors. The actor prototype design can be started using a CAD model of the actor, commonly implemented in commercial CAD tools like AutoCAD®, 3Dstudio® or any software that could produce *3ds* file format that can be then exported to Xfile format. The visualization and configuration of societies is a very important feature of ABAS WorkBench, since it enables saving time in the design of ABAS systems. This is because the designed actors can be emulated, arranged and configured as it would be in the real implementation of the assembly system. Emulated actors and societies, in the same way that real actors and societies, need to be deployed in a platform where they can interact with each other. This platform has been implemented in the ABAS Viewer tool, presented in the next section. Even though the ABAS WorkBench needs a platform, be it ABASViewer or any other that could be developed by a third party, the designed societies can be saved and loaded as projects to enable running multiple design sessions for one system, even if no platform (ABAS Viewer) is connected. The implementation of ABAS WorkBench is divided in four packages, which are connected with each other as shown in Figure 10. The *Emulated-Actor Package* contains all classes used to represent an actor in the ABAS WorkBench emulation environment. In general an emulated actor is an instance of the class AWBActorEmu inherited by definition from Actor-Blue-Print class (Cactor). The *Actor-Prototype-Editor Package* is composed of classes that are used for modeling actors and products prototypes. These classes provide graphic user interfaces for introducing, erasing and modifying actor attributes. This package makes use of the 3D-Interface package for representing the actor attributes in tree dimensional view in the screen in order to have a better understanding of the actor model. The *Society-Editor Package* contains the classes used to provide the necessary tools for editing and visualizing actor societies. This package makes use of the 3D-Interface package in order to show in three-dimensional view the emulated actor societies. This assists the ABAS designer to have a better understanding of the functionality of the system such as type of connections between actors, their position, and orientation related to other actors, along other properties. The *3D-interface Package*, as already mentioned, is used by the other packages. This

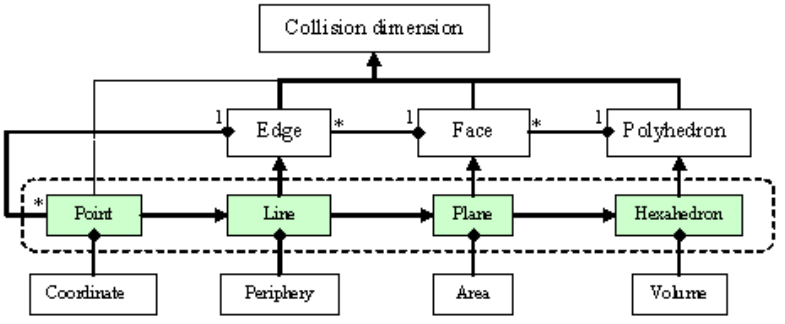


Fig. 9. Collision Dimension class hierarchy

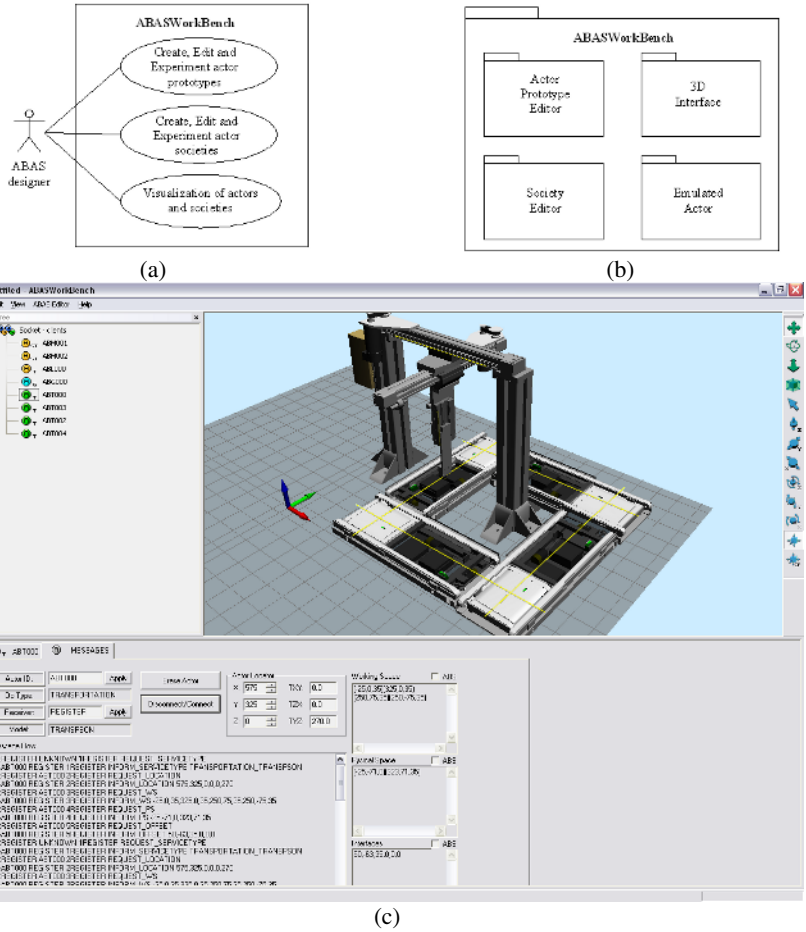


Fig. 10. ABAS WorkBench: a) Use Case diagram; b) the main software packages; and c) A screenshot including the messages traffic.

package has classes that belong to the subset of classes of Direct3D software development kit from Microsoft®; these classes access the hardware resources in order to create three-dimensional views.

4.2 ABAS Viewer

ABAS Viewer incorporates many features of typical Actor-Based Assembly Systems. The main goal of this tool is to monitor the performance of ABAS systems. Because limitations in hardware processing may exist, required algorithms such as recruiting, clustering, broadcasting and loading assembly processes cannot reside in many physical actor implementations. ABAS Viewer provides all logic needed for actor

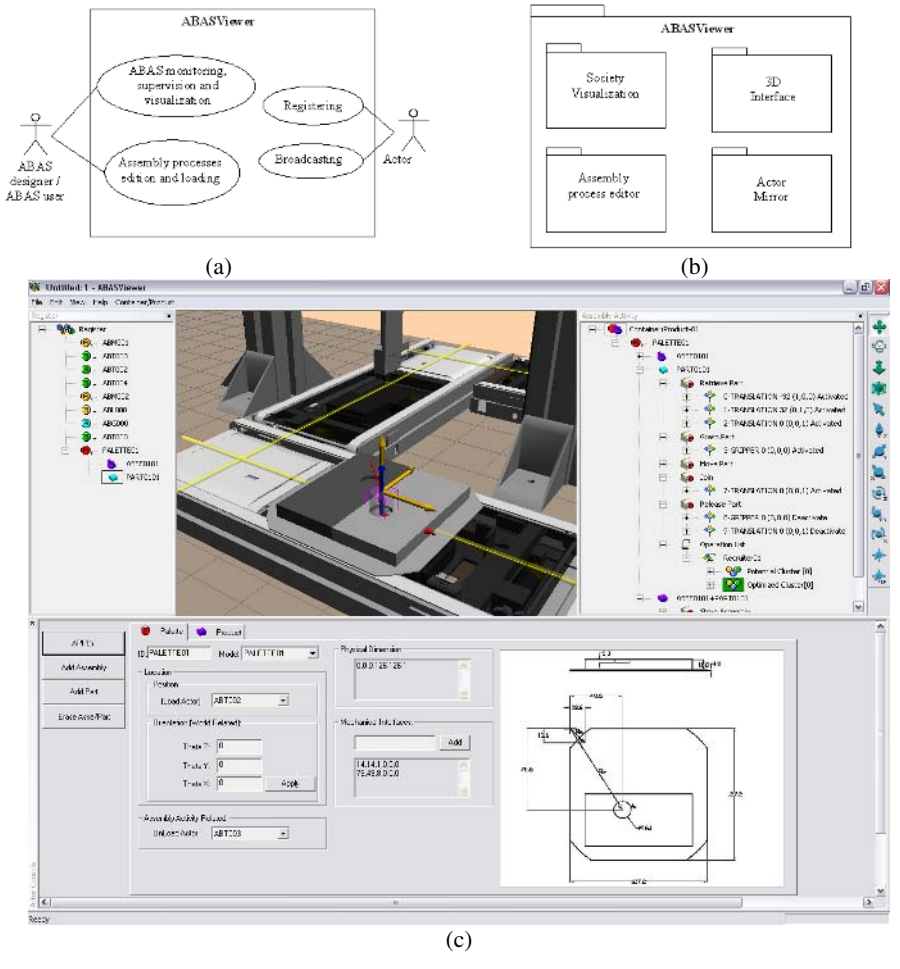


Fig. 11. ABAS Viewer: a) Use Case diagram; b) the main software packages; and c) a screenshot illustrating the different components of an ABAS implementation.

societies to operate and perform assembly processes. Figure 11 shows the Use Case diagram of ABASViewer where not only the user interacts with the system but also the actors, either if they are emulated (created from ABAS WorkBench tool) or real. The user is able to visualize ABAS since this tool uses Direct3D. When all actor attributes are received by the register when certain actor populates the society, those attributes are visualized in the main GUI. The user can also edit and load assembly processes as well as start the execution of them. The actor, on the other hand, uses the ABASViewer to register and to request for communication channels (message transporters) according to FIPA standards. The ABASViewer register is equivalent to the FIPA agent-directory-service.

The ABASViewer tool is divided in four packages shown in Figure 11. The *Society-Visualization Package* contains the classes needed to visualize societies, whether emulated (created from ABAS WorkBench tool) or real indistinguishably. Among other properties the *Society-Visualization Package* has classes for the register creation and GUIs. The *Actor-Mirror Package* has the necessary classes used to store the attributes received from an actor when being registered. The *Assembly-Process-Editor Package* has the GUIs needed to define assembly processes in the system; this package has also the classes for recruiter creation that implement the algorithms for cluster creation and execution.

5 Conclusions and Lessons Learned

A fundamental feature of the approach followed for the implementation of the ABAS tools, without which the development would have been virtually impossible, is the use of established standards, architectures and technologies. Primarily, the use of ABAS architecture has been fundamental in guiding the design of the systems. This experience is highly encouraging for following architecture-based approaches in future work. Moreover, the use of technologies from other domains have helped enrich the usefulness of the tools. In particular, established techniques such as 3D visualization and GUI-based data entry, which are widely used in many other domains, facilitate the use of the domain-specific tools. Thus, the use of the tools is accessible to a wide range of potential users, rather than a reduced set of tool experts. Finally, and within the specific domain of assembly system design, the authors have recognized the value of modeling and emulating the behavior of intelligent physical agents in the design stages of new assembly systems. Utilizing this approach, the designer can focus on the configuration and control aspects of design without making use of a mechanical setup. The configuration of the mechatronic devices can be done concurrently or even after the design has been validated.

References

1. Bollinger, J., et al. Visionary Manufacturing Challenges for 2020, National Research Council Report, National Academy Press, Washington, D.C. 1998.
2. Colombo, A.W., Martinez Lastra, J.L. An Approach to Develop Flexible & Collaborative Factory Automation Systems (FLEXCA). Proceedings of the 4th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering, pp. 549-554. 2004

3. Martinez Lastra, J.L.: Reference Mechatronic Architecture for Actor-based Assembly Systems. Doctoral dissertation No. 484. Tampere University of Technology. ISBN 952-15-1210-5. 2004.
4. Mařík, V., Vrba, P., Macurek, F. Agent-based Solutions for Manufacturing. In: The IEE Control & Automation Professional Network event – Operational Excellence in Consumer Packaged Goods. Cambridge, England. 2003.
5. Mařík, V. Industrial Application of the Agent-based Technology. In Proceedings of 11th IFAC Symposium on Information Control Problems in Manufacturing. Salvador do Bahia, 2004.
6. Parunak, V. Industrial and Practical Applications of DAI. In: Multiagent System A modern approach to distributed artificial intelligence. Ed. G. Weiss. The MIT Press. 1999.
7. Chung, K., Wu, C. Dynamic Scheduling with Intelligent Agents: An Application Note. Metra Application Note 105, Metra, Palo Alto, Ca, 1997.
8. Overgaard, L., Petersen, H., Perram, J. Motion Planning for an Articulated Robot: A Multi-Agent Approach. In Proceedings of Modeling Autonomous Agent in a Muti-Agent World. pp. 171-182, Odense University. 1994.
9. Harrison, R., Colombo, A.W.: Collaborative Automation from Rigid Coupling towards Dynamic Reconfigurable Production Systems. In Proceedings of the 16th IFAC World Congress. 2005.
10. Valckenaers, P. Tutorial on Multi-agent Manufacturing Control. In Proceedings of the 1st IEEE International Conference on Industrial Informatics. Banff. 2003.
11. Lopez Torres, E.: Multi Agent-based Configuration and Visualization Tools for ABAS. Master of Science Thesis. Tampere University of Technology. 2004.
12. Zachman, G. Minimal Hierarchical Collision Detection. Department of computer graphics and virtual reality. University Bonn, Germany. 2002.
13. Eberly, D. Dynamic Collision using Oriented Bounding Boxes, Magic software Inc. 2002.

MAS Methodology for HMS

Adriana Giret, Vicente Botti, and Soledad Valero

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Spain
46022 Valencia, Spain
Phone: +34 96 387 7000
{agiret, vbotti, svalero}@dsic.upv.es

Abstract. Developments in Holonic Manufacturing Systems (HMS) have been reported in three main areas: architectures, algorithms, and methodologies for HMS. Despite the advancements obtained in the first two areas the methodologies for HMS have not received great attention. To date, many of the developments in HMS have been conducted in an almost "empirical way", without design methodology. There is a definite need to have methodologies for HMS that can assist the system designer at every development steps. This methodology should also provide clear and unambiguous analysis and design guidelines. To this end, in this work we present a Multi Agent Methodology for HMS analysis and design.[1]

1 Introduction

In the last ten years, an increasing amount of research has been devoted to holonic manufacturing (HMS) over a broad range of both theoretical issues and industrial applications. We can divide these research efforts into three groups [1]: (i) Holonic Control Architectures, (ii) Holonic Control Algorithms and (iii) Methodologies for HMS. In spite of the large number of developments reported in the first two areas (for a detailed study see [1]), there is very little work reported on Methodologies for HMS. In [2], *a formal specification approach for HMS control* is presented, but it is still in a developmental stage. There are no defined development phases, and no detailed descriptions to explain how to model issues such as cooperation in the holarchy, holon autonomy and system flexibility. In [3], it is proposed an agent organization to model each holon/holarchy that is independent of any holon architecture. However, it is focused only on the holarchy definition and does not define the development phases.

There is a definite need to have methodologies for holonic systems [1], that are based on software engineering principles in order to assist the system designer at each stage of development. This methodology should provide clear, unambiguous analysis and design guidelines. We believe that methodologies from the Multi Agent Technology (MAS) are good candidates for modeling HMS due to the following: the similarities between the holonic and the agent approaches, the wide

[1] This work is partially supported by research grants TIC2003-07369-C02-01 from the Spanish Education Department and CICYT DPI2002-04434-C04-02.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 39–49, 2005.

use of agents as the implementation tool for holonic systems, and the availability of complete MAS Methodologies. However, there are some extensions that must be included in a MAS methodology to be able to model the HMS requirements in a proper way: holon recursive structure, system abstraction levels, HMS specific guidelines, and a mixed top-down and bottom-up development approach.

In this work we present a MAS Methodology for HMS analysis and design. Section 1, introduces the Abstract Agent notion to model the holon recursive structure. Section 2, lists the requirements for a methodology for HMS and Section 3, presents it. Finally, in Section 4, we summarize the conclusions and future works.

2 Abstract Agent and Holon

The HMS consortium has defined the following holon characteristics and holonic concepts [4]:

- Holon - an autonomous and cooperative building block of a manufacturing system for transforming, transporting, storing and/or validating information and physical objects. The holon consists of an information processing part and often a physical processing part. A holon can be part of another holon.
- Autonomy - the capability of a holon to create and control the execution of its own plans and/or strategies (and to maintain its own functions).
- Cooperation - the process whereby a set of holons develops mutually acceptable plans and executes them.
- Self-organization - the ability of holons to collect and arrange themselves in order to achieve a production goal.
- Holarchy - a system of holons that can cooperate to achieve a goal or objective. The holarchy defines the basic rules for cooperation of the holons and thereby limits their autonomy.

An agent is an autonomous and flexible computational system that is able to act in an environment [5].

Holons and agents are very similar concepts (for a detailed comparison of these two notions see [6]). In [6], we pointed out that the recursive structure is the only holon property that is not presented as such in the agent definition. To cope with this limitation, in [7] we proposed the Abstract Agent notion as a modeling artifact for autonomous entities with recursive structures. The Abstract Agent extends the traditional agent definition adding a structural perspective to the agent concept: "*... an Abstract Agent can be an agent; or it can be a MAS made up of Abstract Agents ...*".

The Abstract Agent is an attempt to unify the concepts of holons and agents and to simplify and close the gap between holons and agents in the analysis and design steps. This will make it easer to translate the modelling products that are obtained from methodologies for HMS into coding elements for the implementation of the holonic system. Thanks to the integration of the holon recursive property into an Abstract Agent, the Abstract Agent is useful not only for HMS

but for the modeling of complex systems as well. An Abstract Agent that acts in organizational structures can encapsulate the complexity of subsystems (simplifying representation and design) and can modularize its functionality (providing the basis for integration of pre-existing Multi Agent Systems and incremental development). The Abstract Agent facilitates the modelling of organization of organizations (as well as, Multi Agent Systems of Multi Agent Systems).

3 Requirements of a Methodology for HMS

Manufacturing requirements impose important properties on HMS [4]. These properties define functional attributes and specific requirements for the HMS structure and the HMS development process which must be considered in the methodology. We have defined a HMS methodology requirements list based on the study of the developments reported in HMS and on our experience with software methodologies:

1. Manufacturing control systems require autonomous entities to be organized in hierarchy and heterarchy structures [4].
2. Manufacturing control units require a routine-based behavior that is both effective and timely [8].
3. A methodology for HMS should lead straight-forward from the control task on a factory resource or factory function to autonomous entities [8,4].
4. A methodology for HMS should define a development process that is guided by abstraction levels, and should also provide modeling artifacts, tools and guidelines to manage this process.
5. A methodology for HMS should define a mixed top-down and bottom-up development process.
6. A methodology for HMS should integrate the entire range of manufacturing activities (from order booking through design, production, and marketing) to model the agile manufacturing enterprise [4].

Bearing these requirements in mind we have studied software engineering methodologies which are best suited for problems of this kind. This study has demonstrated that MAS methodologies are good candidates to work with. To this end, we have defined a MAS methodology for HMS which is based on INGENIAS [9] (a complete MAS methodology that has good performance in the development of complex systems). Our approach attempts to satisfy the requirements listed in this section.

4 Methodology

In this section, we present a MAS methodology for HMS analysis and design that is based on the Abstract Agent notion. Every software engineering methodology must define and provide notation, tools, and a development process. In the followings sub-section, we present the notation and the development process of our methodology. The development tools for this methodology will be available in the near future. We use Abstract Agent and holon as similar notions [6].

4.1 Notation

In our approach, the HMS is specified by dividing it in more specific characteristics that form different *views* of the system. These views are defined in terms of MAS technology; therefore, we talk about agents, roles, goals, beliefs, organizations, etc. The views can be considered as general MAS models that can also be applied to other domains. The way in which the views (models) are defined [10,11] is inspired by the INGENIAS methodology. The extensions we have made to the INGENIAS meta-models deal with the following: the addition of the Abstract Agent notion and the properties to model real-time behaviours [12], the redefinition of some relations to conform to the new modeling entities and the dependencies between them. These extensions are motivated by requirements 1 and 2 of Section 3. Here we summarize the models:

- The *agent model* is concerned with the functionality of each Abstract Agent: responsibilities and capabilities.
- The *organization model* describes how system components (Abstract Agents, roles, resources, and applications) are grouped together.
- The *interaction model* addresses the exchange of information or requests between Abstract Agents.
- The *environment model* defines the non-autonomous entities with which the Abstract Agents interacts.
- The *task/goal model* describes relationships among goals and tasks, goal structures, and task structures.

Figure 1 shows some graphical notations of our methodology. The next section shows some example diagrams in which the usage of these notations in the different models is illustrated.

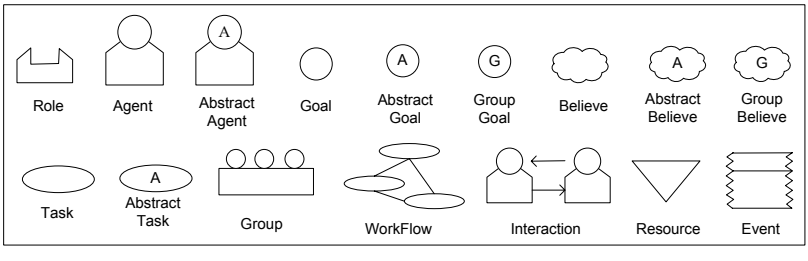


Fig. 1. Some graphical notations of the methodology

4.2 The Development Process

The development process of our methodology provides the HMS designer with clear and HMS-specific modeling guidelines. It also provides complete development phases for the HMS life cycle. The development process is motivated by requirements 3, 4, 5 and 6 of Section 3. In this section, we present the specification of the development process using SPEM diagrams [13]. We also, illustrate

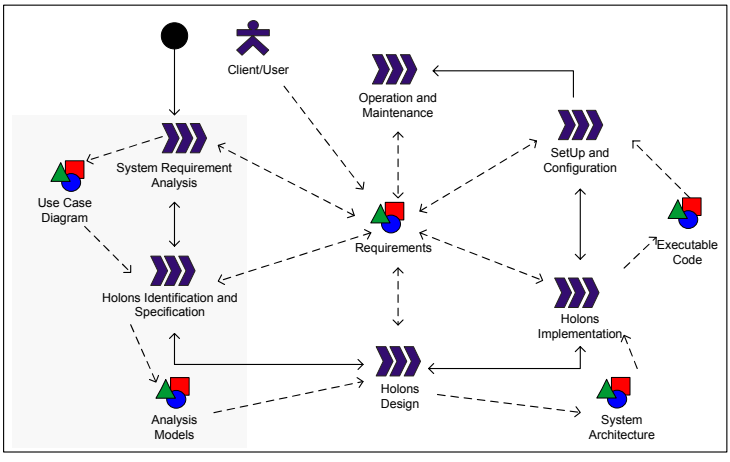


Fig. 2. Stages of the methodology

the development process with modeling diagrams from a Ceramic Tile Factory case study[2]. The tile factory is divided into departments each of which is in charge of a specific function (marketing, tile design, production planning, factory, raw material warehouse, finished tile warehouse, etc.). The tile production process is as follows: the clay is obtained, mixed, refined, dried, pressed or extruded, decorated/glazed and baked in ovens known as kilns. The HMS for the Tile Factory must: (i) integrate the different departments of the company, (ii) arrange factory resources for both on-demand and stock production orders, and (iii) automate resources and processes controls at different levels in the company.

The development stages are presented in Figure 2. The first stage, *System Requirements Analysis* and the second stage *Holons Identification and Specification* define the analysis phase of our approach. The aim of the analysis phase is to provide high-level HMS specifications from the problem *Requirements*, which are specified by the *Client/User* and which can be updated by any development stage. The analysis adopts a top-down recursive approach. One advantage of a recursive analysis is that its results, i.e the *Analysis Models*, provide a set of elementary elements and assembling rules. The next step in the development process is the *Holons Design* stage which is a bottom-up process to produce the *System Architecture* from the *Analysis Models* of the previous stage. The aim of the *Holons Implementation* stage is to produce an *Executable Code* for the *SetUp and Configuration* stage. Finally maintenances functions are executed in the *Operation and Maintenance* stage.

In the analysis phase (Figure 3a), the designer must specify the HMS in terms of the models presented in Section 4.1 and the UML *Use Case Diagrams*. This is a top-down, recursive, incremental process. The main goal of the analysis phase is to identify the constituent holons and to provide an initial holon specification.

[2] The case study requirements where defined in a joint research project between the GTI-IA group and the CIGIP group of the Polytechnic University of Valencia.

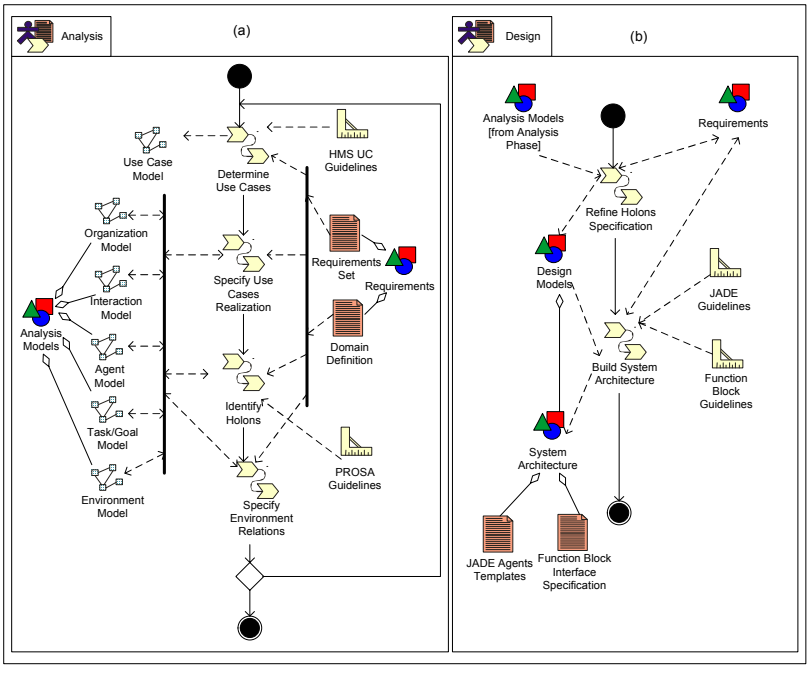


Fig. 3. Analysis and design phases

The designer must produce the *Analysis Models* from the *Requirements Set* and the *Domain Definition*. Each iteration of the analysis phase identifies and specifies holarchies of different levels of recursion (holons made up of holons). The first iteration identifies an initial holarchy, which is made up of holons that cooperate to fulfil the global system requirements. At the end of every iteration, the designer must analyze each holon in order to figure out the advantages of decomposing it into a new holarchy. In this way, each new iteration will have as many concurrent processes as constituent holons of the previous iteration that was decided to decompose. This process is repeated until every holon is completely defined and there is no need for further decompositions. The working definitions of an iteration is as follows.

- The first step is to *Determine Use Cases* by building a *Use Case Model* from the system *Requirements*. We have defined the *HMS UC Guidelines* to help the designer to identify domains cooperations [14] and the system goals as use cases. Use cases can be considered as simpler sub-problems that taken together define the entire system. Figure 4a shows an initial Use Case Diagram of the Tile Factory which is a work product of the first iteration of the analysis phase.
- The use cases of the previous step are analyzed in the step *Specify Use Cases Realization*. Every use case is represented as an Abstract Agent and the interaction and relationships among them are modeled by building *Interaction*

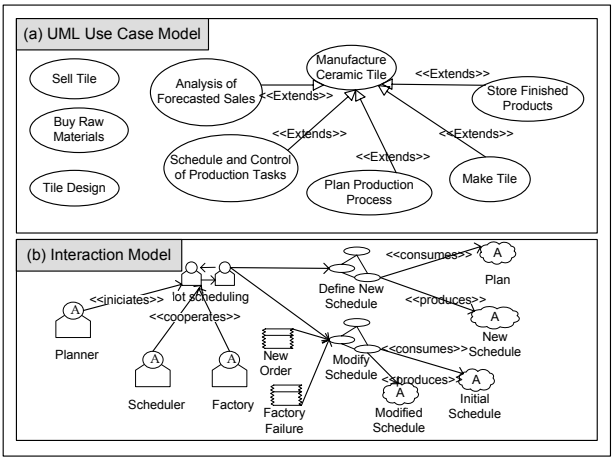


Fig. 4. a) UML Use Case Diagram of the Ceramic Tile Factory. b) An Interaction diagram for scheduling of factory tasks.

Models and *Organization Models*. Figure 5a illustrates the corresponding Organization Model obtained from the Use Case model of Figure 4a.

- In the third step, *Identify Holons*, the designer works with the work products of the previous step, the system *Requirements*, and the *PROSA Guidelines* to identify any new Abstract Agent and to categorize the identified Abstract Agents. The *PROSA Guidelines* are defined based on PROSA types of holons [15]. Some sample rules for identifying holons are: (i) Each production means (a factory, a department, a shop, machine, conveyor, pipeline, component, tool, tool holder, personnel, etc.) and the information processing that controls it, is modeled as an Abstract Agent; (ii) Each product definition or recipe is modeled as an Abstract Agent; (iii) Each task in the manufacturing system (customer order, make-to-stock order, prototype-making order, order to maintain and repair resources, ect.), is represented as an Abstract Agent. Based on these rules the designer must refine both the *Organization* model and the *Interaction* model by adding new or modified relations and interactions among holons in the cooperation domains. Figure 4b shows the Interaction Model to define a new schedule for an order, or to modify a previously defined schedule due to factory failures or new orders. The *Agent Model* (Figure 5b) is built to specify holon capabilities and responsibilities in terms of tasks and goals which are described in detail in the *Task/Goal Model*.
- The *Environment Model* is built in the fourth step, *Specify Environment Relations*, to represent non-autonomous domain entities with which the holons have to work.

In the design phase the initial system architecture, i.e. *Analysis Models*, must be completed with details of the target implementation platform (Figure 3b). This phase is divided into two steps.

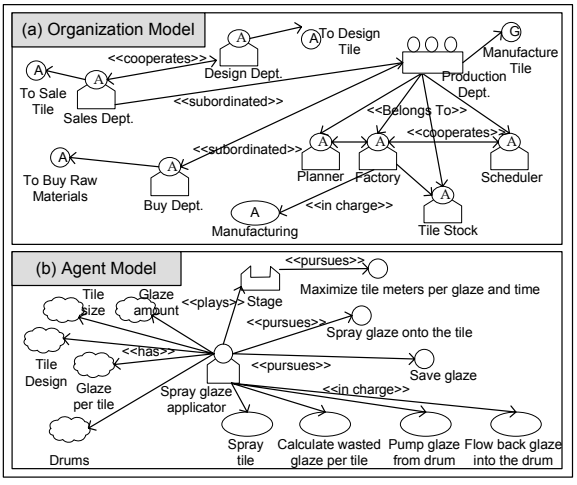
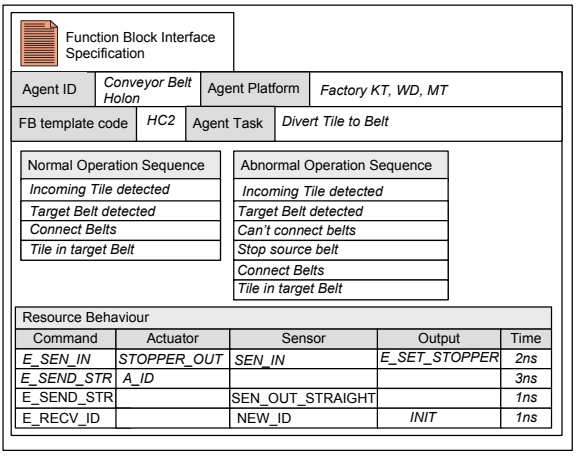


Fig. 5. a) An Organization Diagram of the Ceramic Tile Factory. b) The Agent Model of the *Spray glaze applicator* holon.

- The first design step, *Refine Holons Specification*, is dedicated to complete analysis models without taking into account platform modeling issues. The designer must focus on the "atomic" holons of the previous phase in order to complete their definitions. The *Agent Model* must be revised to include the internal execution states of the holon and their transitions. The *Task/Goal Model* must be analyzed to ensure that each agent goal has a corresponding task that pursues it. Pre and post conditions have to be identified for every modelled task. The *Environment Model* must be detailed to include the resource attributes and the agents perceptions in terms of application events. Once every atomic holon is completely specified, the designer must move up to the nearest abstraction level in the holarchy structure, i.e., to the cooperation domain in which the given holon interacts. Dependencies among cooperation domains/holarchies are refined in the *Organization Model*. The *Interaction Model* is enhanced with preconditions, task executions and effects on the environment and on interacting holons. This bottom-up process must be repeated until there is no higher cooperation domain in the *Analysis Models*.
- The last design step is *Build System Architecture*. Our approach defines design guidelines to implement the HMS as proposed by Christensen in [16]. For high-level control (intra-holon information processing and inter-holon cooperation), our methodology provides design guidelines for JADE [17]. For the low level control (physical operations), our methodology provides design guidelines for function blocks (IEC 61499 series of standards) [18]. The work product of this step is the *System Architecture* composed of the *Design Models* (built in the previous step), the *JADE Agent Templates* and the *Function Block Interface Specification*. A *JADE Agent Template* is produced for each



Function Block Interface Specification

Agent ID	*Conveyor Belt Holon*	Agent Platform	*Factory KT, WD, MT*
FB template code	*HC2*	Agent Task	*Divert Tile to Belt*

Normal Operation Sequence
Incoming Tile detected
Target Belt detected
Connect Belts
Tile in target Belt

Abnormal Operation Sequence
Incoming Tile detected
Target Belt detected
Can't connect belts
Stop source belt
Connect Belts
Tile in target Belt

Resource Behaviour

Command	Actuator	Sensor	Output	Time
E_SEN_IN	*STOPPER_OUT*	*SEN_IN*	*E_SET_STOPPER*	*2ns*
E_SEND_STR	*A_ID*			*3ns*
E_SEND_STR		SEN_OUT_STRAIGHT		*1ns*
E_RECV_ID		NEW_ID	*INIT*	*1ns*

Fig. 6. A Function Block Interface Specification for the task *Divert Tile to Belt* of the *Conveyor Belt Holon*

```
public class GlazeLineManager extends Agent {
// The type of the applicator to find
private String typeApplicator;
// The list of known Glaze Applicators
private AID[] glazeApplicator;
// Put agent initializations here
protected void setup() {
....

//Add a SearchSprayGlazeBehaiour that schedules a request to Glaze Applicators every minute
addBehaviour(new SearchSprayGlazeBehaiour(this, 60000) {
protected void onTick() {
// Update the list of glaze applicators
DFAgentDescription template = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("spray-glaze-applicator");
template.addServices(sd);
try {
DFAgentDescription[] result = DFService.search(myAgent, template);
glazeApplicator = new AID[result.length];
for (int i = 0; i < result.length; ++i) {
glazeApplicator[i] = result.getName();
}
}
catch (FIPAException fe) {
fe.printStackTrace();
}
// Perform the request
myAgent.addBehaviour(new RequestPerformer());
}
} );
...
```

Fig. 7. The JADE fragment code for the behaviour of the *Glaze Line Manager* to search for a *Spray Glaze Applicators* in a glaze line

agent in the *Design Models*. The *Function Block Interface Specification* is produced for the physical processing part of each agent representing physical processes, equipment or machines. The *JADE Agent Template* contains JADE specific characteristics such as agent identifiers, agent behaviours, agent communication and services. The *Function Block Interface Specification* contains a table so that there is an ordered list of corresponding physical

device commands and responses for every agent physical action (task). Figure 6 shows an example *Function Block Interface Specification* for the task *Divert Tile to Belt* of the *Conveyor Belt Holon*.

From the *System Architecture* the *Holons Implementation* phase produces the *Executable Code* for the HMS. In this phase the programmer has to implement the information processing part of each JADE-agent and the physical processing part of each agent representing physical processes, equipment or machines. For the first task the designer may use the JADE programmers guide [17], and for the second task he may use the programmers guide defined by the standard IEC 61499 [18]. Figure 7 shows an example JADE fragment code for the *Glaze Line Manager* holon. To implement the intra-holon communication the programmer can implement a blackboard system [19] or a special management service interface function block [20]. Configuration activities are carried out in the *SetUp and Configuration* phase to deploy the HMS at the target destination. Finally in the *Operation and Maintenance* phase maintenance activities are performed. In the case of new requirements, a new development process must be initiated.

5 Conclusion

In this work, we have presented the notation and development process of a MAS methodology for HMS. In our approach the HMS is specified by dividing it in more specific characteristics that form different *views* of the system. These views are define based on INGENIAS (a proven MAS methodology for complex systems) [9]. We have extended the INGENIAS models to be able to model the HMS requirements properly. These extensions include the notion of Abstract Agent [7] and the properties to model real-time behaviours [12]. They also include the redefinition of some relations to conform to the new modeling entities and the dependencies among them. The development process we have proposed is a mixed top-down and bottom-up approach. The aim of the analysis phase is to provide high-level HMS specifications from the problem *Requirements*, which are specified by the *Client/User* and which can be updated by any development stage. The analysis adopts a top-down recursive approach. One advantage of a recursive analysis is that its results, i.e the *Analysis Models*, provide a set of elementary elements and assembling rules. The next step in the development process is the *Holons Design* stage which is a bottom-up process to produce the *System Architecture* from the *Analysis Models* of the previous stage. The aim of the *Holons Implementation* stage is to produce an *Executable Code* for the *SetUp and Configuration* stage. Finally maintenances functions are executed in the *Operation and Maintenance* stage. Our approach provides HMS-specific guidelines to help the designer in every development step.

We are currently working on the evaluation of our methodology with industrial case studies; for example: the Ceramic Tile Factory presented in this work, and an Assembly and Supplier Company of Automobile Parts. We are also working on CASE tools for our methodology.

References

1. McFarlane, D., S., B.: Holonic Manufacturing Control: Rationales, Developments and Open Issues. In: Agent-Based Manufacturing. Advances in the Holonic Approach. Springer-Verlag (2003) 301–326
2. Leitao, P., Restivo, F.: An Approach to the Formal Specification of Holonic Control Systems. Holonic and Multi-Agent Systems for Manufacturing. LNAI 2744. ISSN 0302-9743 (2003) 59–70
3. Fischer, K., Schillo, M., Siekmann, J.: Holonic Multiagent Systems: A Foundation fo Organisation of Multiagent Systems. Holonic and Multi-Agent Systems for Manufacturing. LNAI 2744. ISSN 0302-9743 (2003) 71–80
4. HMS, P.R.: HMS Requirements. HMS Server, http://hms.ifw.uni-hannover.de/ (1994)
5. Wooldridge, M., Jennings, N.R.: Intelligent Agents - Theories, Architectures, and Languages. Lecture Notes in Artificia Intelligence, Springer-Verlag. ISBN 3-540-58855-8 **890** (1995)
6. Giret, A., Botti, V.: Holons and Agents. Journal of Intelligent Manufacturing **15** (2004) 645–659
7. Giret, A., Botti, V.: Towards an Abstract Recursive Agent. Integrated Computer-Aided Engineering **11** (2004) 165–177
8. Bussmann, S., Jennings, N., Wooldridge, M.: Multiagent Systems for Manufacturing Control. A design Methodology. Springer Verlag (2004)
9. Pavon, J., Gomez, J.: Agent Oriented Software Engineering with INGENIAS. 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003) : V. Marik, J. Mller, M. Pechoucek:Multi-Agent Systems and Applications II, LNAI 2691 (2003) 394–403
10. Giret, A., Botti, V.: Towards a Recursive Agent Oriented Methodology for Large-Scale MAS. Agent-Oriented Software Engineering IV. **LNCS 2935** (2004) 25–35
11. Giret, A., Botti, V.: On the definition of meta-models for analysis of large-scale MAS. In: Multiagent System Technologies. LNAI 3187 (2004) 273–286
12. Julian, V., Botti, V.: Developing real-time multiagent systems. Integrated Computer-Aided Engineering **11** (2004) 135–149
13. OMG, O.M.G.: Software Process Engineering Metamodel Specification Version 1.0. http://www.omg.org/docs/formal/02-11-14.pdf (2002)
14. Fletcher, M., Garcia-Herreros, E., Chritensen, J., Deen, S., Mittmann, R.: An Open Architecture for Holonic Cooperation and Autonomy. Proceeding of HoloMAS'2000 (2000)
15. 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
16. Christensen, J.: HMS/FB Architecture and Its Implementation. In: Agent-Based Manufacturing. Advances in the Holonic Approach. Springer Verlag (2003) 53–88
17. JADE: Java Agent DEvelopment Framework. http://jade.tilab.com/ (2005)
18. IEC: International Electrotechnical Commission: Function Blocks, Part 1 - Software Tool Requirenments.PAS 61499-2. (2001)
19. McFarlane, D., Kollingbaum, M., Matson, J., Valckenaers, P.: Development of algorithms for agent-oriented control of manufacturing flow shops. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. (2001)
20. Fletcher, M., Brennan, R.: Designing a holonic control system with iec 61499 function blocks. In: Proceedings of the International Conference on Intelligent Modeling and Control. (2001)

Probabilistic Holons for Efficient Agent-Based Data Mining and Simulation

Arndt Schwaiger and Björn Stahmer

DFKI GmbH, Stuhlsatzenhausweg 3,
66123 Saarbrücken, Germany
{Arndt.Schwaiger, Bjoern.Stahmer}@dfki.de

Abstract. Multiagent systems (MAS) provide a useful approach to distributed problem solving whereby the growing complexity of robust and scalable systems increases the requirements on efficient structures and organisations. A promising approach to organisation of MAS is the theory of holonic multiagent systems (HMAS), which allows different kinds of recursive agent groupings (holons) whose members temporarily collaborate to solve a certain sub-problem. We present a holonic multiagent system based on a probabilistic agent architecture which is efficient for agent-based distributed data mining and simulation. We illustrate its ability to build different kinds of high-performance holons which are useful for different simulation aspects. The holons are built using cloning and merging of probabilistic networks representing the behaviour and knowledge of the agents. Finally, we show their benefits within the project SimMarket concerning supermarket simulation based on a framework for probabilistic HMAS.

1 Introduction

The idea of multiagent systems (MAS) in general is that the solution of a given problem emerges from interaction between autonomous agents whereas the capability of the whole MAS exceeds the capabilities of the individual agents [2][3]. Multiagent systems are a useful approach to distributed problem solving whereby the complexity is reduced to sub-problems which can be solved by specific agents. But the growing complexity of robust and scalable systems increases the requirements on efficient structures and organisations of multiagent systems [4]. A promising approach to organisation of agent societies is the theory of holonic multiagent systems (HMAS) [1], which provides a methodology for recursive modeling of agent groups and allows dynamic reorganisation during runtime. Holonic multiagent systems have the ability to dynamically reorganise during runtime to adapt to changing goals and environments. Certain classes of goals are only reachable by sets of collaborating agents where the agents give up parts of their autonomy and merge together to agent groups with more efficient structures. In this paper we present an architecture for holonic multiagent systems based on probabilistic agents which is useful for agent-based distributed data mining and agent-based simulation. The knowledge bases of the probabilistic agents consist of sets of Bayesian networks in order to let agents be able

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 50 – 63, 2005.

to reason about their knowledge with uncertainty. Structure and the parameters of the Bayesian networks are learned from real world training data whereas existing domain knowledge is incorporated. This paper focuses on how probabilistic agents could efficiently combine their problem solving capabilities and knowledge bases in order to perform common simulations or to solve global problems. Our approach is to temporarily build holons based on different types of partial or complete cloning and merging of probabilistic networks which is efficient for performing sequences of similar simulation tasks.

In section 2 we give an introduction to holonic multiagent systems according to [1] including three general possibilities to build holons (*holonification)*.Thereafter, we introduce an efficient kind of holonification based on the concept of partial and complete agent cloning and merging.

In section 3, we present a probabilistic agent architecture based on Bayesian networks. Using the architecture, we design a probabilistic holonic multiagent system (section3.1). In sections 3.1.1 and 3.1.2, we give two examples of probabilistic agents which have to collaborate to perform a common simulation task, in order to demonstrate possible advantages of probabilistic holons based on network merging. In section 3.2, we introduce three different merging types for probabilistic networks, which are useful for different modeling and simulation tasks.

In section 4, we describe a framework for holonic probabilistic multiagent systems and present its usage and advantage within the project SimMarket concerning agent-based supermarket simulation.

2 Holonic Multiagent Systems (HMAS)

The emergent problem solving behaviour of multiagent systems consists of interaction of autonomous agents which are able to solve local problems whereas the idea is that the agents collaborate to solve a given global problem. Although interesting results have been presented using the multiagent approach, efficiency suffers from interaction and communication activities in complex systems. Thus, communication costs have to be reduced. An idea to limit the interaction activities is to adapt some aspects of the *divide and conquer* concept where the global problems are divided in a hierarchy of sub-problems which are distributed to decentralised problem solvers. To adopt the structured problem solving behaviour, we need the realisation of dynamically organised agent societies which is the main idea of *holonic multiagent systems*, where an agent may consist of many *sub-agents* or a group of agents may decide that it is more efficient to solve a certain problem by joining to a *super-agent* that appears as a single agent. An agent that is composed of sub-agents is called a *holon* which is a self-similar or fractal structure that is stable and coherent and that consists of several holons as sub-structures [5], whereby a single agent without sub-agents is defined as *atomic* holon. In order to execute a certain sub task or to solve a certain sub-problem, agents are able to join temporarily to a holon (*holonification*) in order to collaborate more efficiently whereas all sub-agents give up a part of their autonomy. Agents can be members of several holons at the same time. A holon is represented by an individual agent which is called the *head* of the holon. During the holonification process, an

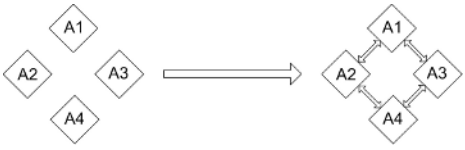


Fig. 1. A holon as a set of autonomous agents

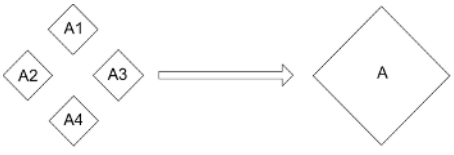


Fig. 2. Autonomous agents merge into one

arbitrary participating agent can be selected or elected as head, whereby all other agents represent the *body* of the holon. It is also possible to explicitly introduce a new agent to represent the holon during its lifetime.

In general there are three different possibilities to model holonic structures within a multiagent system, which differ mainly in the degree of autonomy of the participating agents.

In the first case, a holon consists of a *set of autonomous agents* (Fig. 1), where the participating agents collaborate to reach a common goal. In this manner a holon represents a traditional multiagent system. In addition, the communication costs can be limited by allowing the agents only to interact with agents of the same holon in order to reach the holon's goal.

Fig. 2 illustrates the second possibility to build a holon. All participating agents give up completely their autonomy and *merge into a new agent* that combines all knowledge, capabilities and functionalities of the sub-agents. If the sub-agents use the same representation models for knowledge, capabilities and goals, then the merging process consists of their union in consideration of consistency. Otherwise, if the models are heterogeneous then the merging is intractable. The advantage of that kind of holonification is the complete omission of communication and interaction activities. After the common goal is reached the holon can be terminated and the participating agents are recreated, but in most cases, this recreation process is also costly or even impossible.

The degree of autonomy of the third solution for holonification lies between the presented extremes. The participating agents give up only a part of their autonomy to create a holon (Fig. 3). For efficiency reasons a certain agent is selected as head in order to represent and administer the holon. The competence of the head can range from purely administrative tasks to the authority to give directives to the body-agents and to plan and negotiate for the whole holon based on the sub-agent's plans and goals, and even to remove or insert agents within the holon [1].

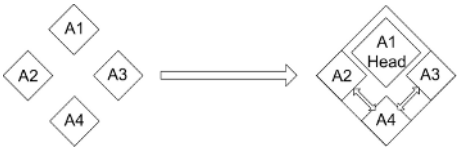


Fig. 3. A holon as a head-guided agent society

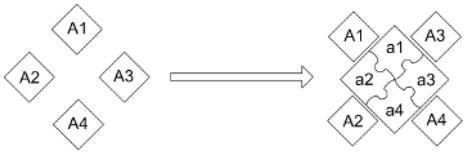


Fig. 4. Holonification based on partial agent cloning and merging

In that kind of *head-guided agent society*, all sub-agents are more or less semi-autonomous agents and thus they do not lose their capability of distributed problem solving.

In general holonic multiagent systems provide a flexible organisational structure which is required by robust and scalable software systems. In addition the efficiency is increased by reduction of communication and interaction activities. But there are also some disadvantages: in case of holons as autonomous agents or head-guided agent societies, communication is still required and in case of no interaction, the agents completely give up their autonomy, the system loses the distributed problem solving behaviour and the merging process of complete agents may be intractable. With intent to combine the advantages of these approaches and to reduce their disadvantages, we present holonification based on *partial agent cloning and merging* (Fig. 4).

In order to solve a certain problem cooperatively, participating agents create a copy of their knowledge and functionality which are relevant for the given problem (partial agent cloning). In the next step, all partial agent copies are merged within a new agent which represents the holon. The advantage is that the agents remain autonomous and are able to solve other problems while the holon only consists of relevant knowledge and functionality. The concept could also be used to design genetic agents in terms of increasing the performance or experience of agents by evolution, i.e. mutant cloning and merging.

3 Probabilistic Agent Architecture

In order to support agent-based distributed data mining and agent-based simulation, we developed a probabilistic agent architecture. A probabilistic agent is defined by $A_{prob} := (TM, FK, PSK)$, where TM is a set $\{T_1,\ldots,T_n\}$ of tasks and methods which the agent is able to perform, FK is the *fact knowledge* containing a set $\{F_1,\ldots,F_p\}$ of facts and PSK is the *probabilistic structural knowledge*. The fact knowledge consists of any kind of information which is available to the agent and provides the basis for the generation of probabilistic structural knowledge which is modeled as a set

$\{BN_1,\ldots,BN_q\}$ of *behaviour networks*. A behaviour network is described by BN := (DMT, SB, DK), where DMT is a set of *data mining tables* generated from fact knowledge, SB is a set $\{B_1,\ldots,B_r\}$ of *Bayesian networks* which are learned based on data represented by the DMT, and DK is *domain knowledge* which is used to define structure, context and semantic of the Bayesian networks. The TM of probabilistic agents includes by default methods for *identifying maximal equal subnets* of Bayesian networks and different kinds of *network merging* [10]. Based on these methods it is possible to build holons using complete and partial agent merging.

3.1 HMAS Based on Probabilistic Agents

In this section we present an approach to holonic multiagent systems based on probabilistic agents, whereby we propose solutions for each kind of holonification illustrated in section 2. Probabilistic agents are efficient for distributed data mining and agent-based simulation, whereby each agent has its own local knowledge which can be combined to global knowledge. In case of agent-based simulation, the agent's knowledge represents its behaviour in given situations.

Using two examples, we want to explain the realisation, usage and advantage of different kinds of holonification using probabilistic agents. Both examples are related to a MAS including *n* probabilistic agents which have the common task to compute a sequence of *s* simulation steps. The main goal is to simulate the probability that a certain agent, namely Agent 1, performs a certain task, which is represented by a random variable *X*, depending on three observable factors *A*, *B* and *C*. The sequence of computations consists of *s* simulation steps at which the observable values of the three factors are able to change. We assume that all other variables in both examples are not observable by any agent of the MAS.

3.1.1 First Example: Two Collaborating Agents

Fig. 5 shows the first example. There are two agents, Agent 1 and Agent 2, which are both members of the MAS. Their knowledge bases consist of simplified probabilistic networks. Observable variables are marked by dotted lines (variables *A*, *B* and *C*).

To solve the global problem of computing the probability distribution of variable *X* depending on *A*, *B* and *C*, Agent 1 will start trying to compute the requested probability distribution. But Agent 1 will fail because variable *X* is dependent on an unobservable variable *G* which is itself dependent on the unobservable variable *D* and the observable variable *C*. Because Agent 1 has no information about *A* and *B*, and *D* is not observable, it is not able to solve the problem by itself and is reliant on other agents. Agent 1 has to start a search for appropriate agents.

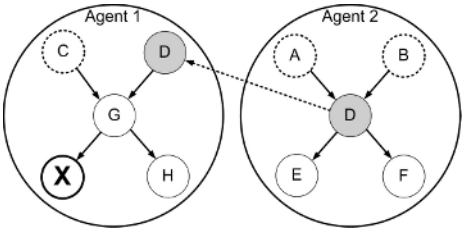


Fig. 5. Two collaborating agents

In our example, we assume that only Agent 2 knows variable *D* which is dependent on variables *A* and *B*.

In case of a **normal MAS**, in worst case, Agent 1 has to ask all other *n*-1 agents if they are able to compute an appropriate probability distribution for variable *D* or have information about *A* or *B*. If asked, Agent 2 will compute the requested values and pass them to Agent 1 which is now able to compute the probability distribution of *X*. Because the task consists of a sequence of *s* computations of the values of *X* with possibly changing values of *A*, *B* and *C*, Agent 1 has to ask all *n*-1 agents for each of the *s* steps (in worst case) for the computation of variable *D*.

In order to limit the communication costs, a holon as **set of $h \leq n$ autonomous agents** could be built whose members must know at least one of the variables *A*, *B*, *C* and *X*. In that case, Agent 1 has to ask only *h*-1 agents for appropriate values for variable *A*, *B* or *D* (in worst case).

In case of a holon as **head-guided agent society**, the process and communication costs could be decreased, if the head has the authority and the knowledge to draw up a plan in which Agent 1 has to ask only Agent 2 to compute the missing values for variable *D*.

In case the holon is built by **complete merging of all participating agents**, all knowledge bases including all probabilistic behaviour nets have to be merged whereas complete merging could be very costly or rather not practicable and the resulting agent may be too complex. However, in many cases complete merging is possible and can result in detection of new knowledge if the merging algorithm discovers relations between variables from different knowledge bases (for more details about our relation merging algorithm see [10]). Fig. 6 shows on the left the result of complete merging of the probabilistic networks of Agent 1 and Agent 2. As highlighted by the dotted arrow, the merging algorithm discovered a new relation between variables *F* and *X*. We call the new information *emergent knowledge*, because it represents knowledge which was discovered by combining the experience of different agents. In order to compute *s*-times the probability distribution of variable *X* depending on factors *A*, *B* and *C* , the holon does not need any further communications between Agent 1 and Agent 2 after the complete merging is finished. In addition, the result will be more precise because of the comprehension of relevant emergent knowledge (edge between variables *F* and *X*).

In order to realise holonification based on **partial agent cloning and merging**, a new agent is created based on partial copies of the agent's knowledge bases which are required for the given problem (Fig. 6 on the right). For example variables *E* and *H* are

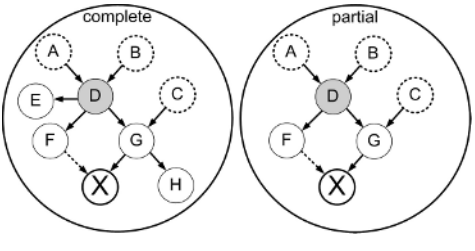


Fig. 6. Complete and partial emergent merging

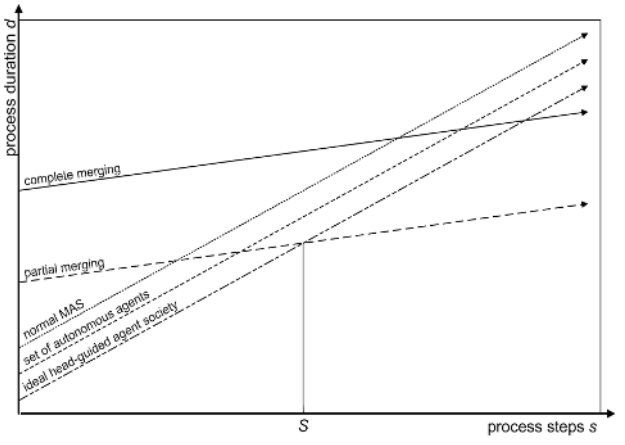


Fig. 7. Comparison of different holon types (Example 1)

omitted, because they have no direct influence on variable *X*. Again, the partial merging process provides the possibility to discover emergent knowledge (in this case the relation between variables *F* and *X*). The advantages over complete agent merging are that only the important parts of the probabilistic behaviour networks are combined in place of complete agents which may contain large sets of behaviour networks that may include themselves sets of Bayesian networks. In addition, the participating agents stay autonomous and are able to solve other problems, because the holon is built of partial copies.

Fig. 7 shows a comparison of the simulation process durations using different holon types. In case of normal MAS, set of autonomous agents and head-guided agent society we assume that Agent 1 keeps in mind that it has to ask only Agent 2 after the first of the *s* computation steps. At the beginning both holons based on merging need more computation time because of the expensive merging processes, but after merging they can concentrate on computing the requested probability distribution for variable *X* by inference of observed values for *A*, *B* and *C* without any further communication. The other holon types need less time for initialisation, but the participant agents have to communicate during all *s* computation steps. Thus, there exist a number S of computation steps, where the holons based on merging – especially the holon based on partial cloning and merging – are more efficient than all other types of holons (see Fig. 7), whereas there is additionally the possibility to discover emergent knowledge during the merging process which can be included in the computation in order to provide more precise results.

3.1.2 Second Example: Three Collaborating Agents

In our second example, we change the knowledge base of Agent 2 and introduce Agent 3 (see Fig. 8). To execute the current task – computing a sequence with *s* steps of the probability distribution of variable *X* depending on changing values on factors *A*, *B* and *C* – the three agents have to collaborate. We assume that all other agents of

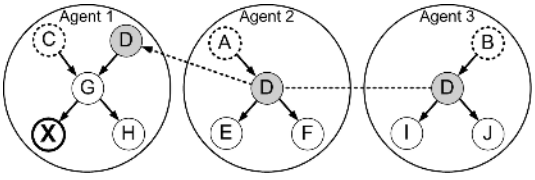


Fig. 8. Three collaborating agents

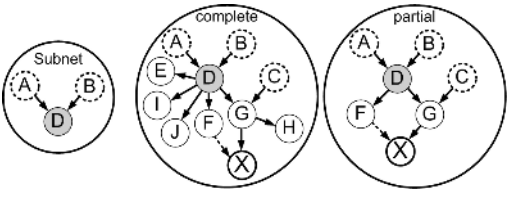


Fig. 9. Subnet, partial and complete merging

the MAS have no knowledge about variables *A*, *B*, *C*, *D* and *X*. Analogous to our first example, Agent 1 has no information about the value of variables *A*, *B* and *D*. Again, Agent 1 is reliant on other agents to finish the computation.

In case of a **normal MAS**, Agent 1 has to ask all *n*-1 agents (in worst case) if they have information about variable *A*, *B* and/or *D*. Agent 1 will find Agent 2 and Agent 3, which both have partial knowledge about *D*, whereas Agent 2 knows the relation between *A* and *D*, and Agent 3 has information about the relation between *B* and *D* (see Fig. 8). In order to be able to compute the value of *X* complex information has to be exchanged between the three agents. The best way would be that Agent 2 and Agent 3 send their relevant subnets $A \rightarrow D$ and accordingly $B \rightarrow D$ to Agent 1 which could merge the subnets into a single subnet (see Fig. 9 on the left) in order to be able to compute the value for *D* depending on *A* and *B*. Thus, Agent 1 can compute the requested probability distribution of *X*. To decide what subnets should be exchanged an algorithm similar to [8] could be used. In worst case this procedure has to be executed for each of the *s* computation steps.

A set of autonomous agents is again able to reduce the communication costs from *n*-1 to *h*-1 requests, while a **head-guided agent society** could limit the communication costs if the head has the authority and knowledge to draw up a plan where Agent 1 has only to ask Agent 2 and Agent 3.

In case of **complete merging**, the merging process is very similar to the first example (see Fig. 9 in the middle). The difference consists in the number of merged networks and existence of the variables *I* and *J* which are not relevant for the computation of *X*. Again, emergent knowledge is discovered (edge $F \rightarrow X$).

The result of **partial cloning and merging** is the same as in the first example, because the relevant subnets are equal (Fig. 9 on the right).

Fig. 10 shows a comparison of the simulation process durations of the different types of holons concerning the second example. The initialisation time and the

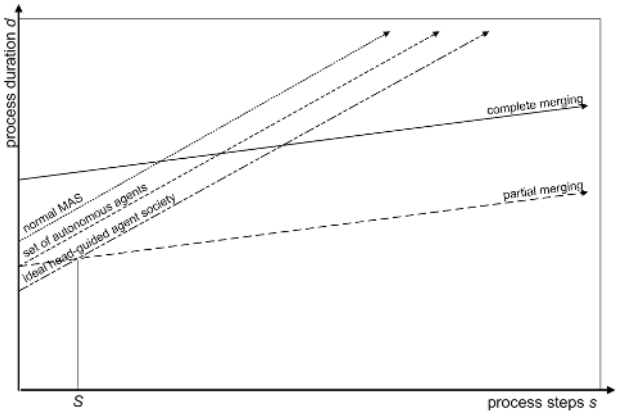


Fig. 10. Comparison of different holon types (Example 2)

duration of a single computation step of normal MAS, set of autonomous agents and head-guided agent society is increased, because exchange and merging of subnets from three agents is more expensive than the exchange of simple values between two agents from example 1. In case of holons based on merging, the initialisation takes also a bit more time, but thereafter each computation step is equal to example 1. Again emergent knowledge is discovered and used during the computation in order to be able to provide more precise results. In example 2, the number of steps *S* is decreased, where the holon based on partial cloning and merging is more efficient than all other holon types.

The partial cloning and merging process may be extensive or intractable in some situations, but in case that there is a large sequence of problems of the same type, the partial merged holons highly reduce the communication costs. Additionally, there is the advantage of emergent knowledge which leads to more precise results in distributed data mining and agent-based simulation. The concept of partial and complete cloning offers the possibility to use the original agents to perform some other tasks at the same time.

3.2 Average, Addition and Relation Merging

In some scenarios it is efficient to merge agents of the same type or agents which contain behaviour networks based on the same structure. Depending on the given problem it is either necessary to create an *average* or *addition* network or to discover *relations* between certain nodes. Fig. 11 illustrates the different cases of merging two agents based on the same structure including nodes *A*, *B*, *C*, *D* and *E*.

On the left an **average network** is generated by "averaging" the node values of both networks, i.e. to average both data mining tables and to relearn the network to discover new relations in order to update the structure. Average merging is useful in agent-based simulation if the user is interested in average behaviour of certain agents.

Addition networks represent the aggregate behaviour of agents (see Fig. 11 in the middle). Addition merging can be realised by adding the values of the data mining or probability tables.

In other situations it is interesting to discover **relations** between certain nodes, e.g. if agent's behaviour is dependent on other agent's nodes of the same type. In this case, the nodes of interest have to be analysed if they are conditionally dependent while the other nodes have to be averaged in an analogues manner as in the average case (see Fig. 11 on the right). The merging process can be realised by joining the data mining tables whereby data columns of the nodes of interest have to be distinguished while the data columns of the average nodes are merged. The domain knowledge of the behaviour networks is used to decide, which node types have to be merged and which have to be distinguished in order to find dependencies. Domain knowledge can be represented as a set of node types and appropriate rules that determine what combinations of node types should be merged or distinguished.

In addition, a **combination of the three merging types** is also possible by adding rules to decide what node type should be merged using what kind of merging.

In general, all kinds of holonification have advantages in certain situations, whereby the most significant difference consists of the trade-off between communication costs, emergent knowledge and extensive merging. We propose a multiagent system design which supports all presented types of holonification as well as agent cloning in order to provide a distributed micro problem solving architecture that is able to temporarily combine agents to holons of a type that is the most efficient for the given sub-problem. Furthermore, the partial and complete agent cloning strategy appears efficient in agent-based simulations. In this case the consistency of all agent clones can be warranted more tractable, because the Bayesian networks are already learned and only used for simulation of the agent's behaviour, whereby emergent knowledge may be discovered and used in holons. In order to increase the performance of future holonifications, the information about emergent knowledge can be stored in the domain knowledge of the original agent.

The domain knowledge of the behaviour networks and the information in the data mining tables can be used to decide if different networks can be efficiently merged and to reduce the costs of the merging process. In addition, we define for each node if it is an *input*, *output* or an *input-and-output* node in order to reduce the search space for new relations. In general the possibility of merging networks is dependent on the general context, semantic of the network nodes (node types), the given problem and temporal aspects of the information and data which are encoded in the data mining tables and in certain time nodes.

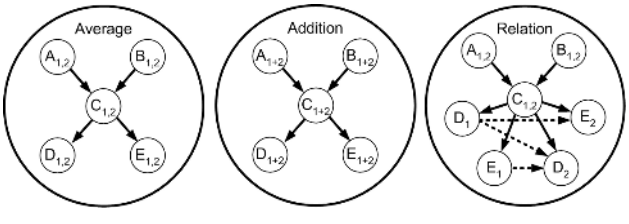


Fig. 11. Average, Addition and Relation Merging

4 Application Supermarket Simulation

In this section we present the project SimMarket which concerns agent-based supermarket simulation based on a multiagent system framework which supports probabilistic agents and different kinds of holonification. First, we give a short introduction into SimMarket and illustrate how the different types of holonification are used within the supermarket simulation.

In order to realise complex multiagent systems especially for agent-based distributed data mining and agent-based simulation, we developed a framework for distributed multiagent systems. The framework supports different kinds of agent groupings as well as all presented types of holonification concerning probabilistic agents (see section 3). In order to handle behaviour networks for modelling probabilistic structural knowledge a complex component for Bayesian networks was implemented which supports complete and partial network cloning and merging. For agent-based simulation or simultaneous applications, agents are able to create an arbitrary number of complete or partial clones. Agents are able to retrieve information from connected web services, components, data bases or data warehouses which have to be registered to the framework. Based on the possibility to retrieve information from knowledge bases and to learn Bayesian networks from data tables we support agent-based distributed data mining.

Based on the framework, a supermarket simulation environment was developed within the scope of project SimMarket whose main research goal is to approach the optimisation of assortment, price and promotion policy of retail stores [6]. In particular, the problem is to find an optimal set of internal changes inside a certain supermarket store from the situation today to a future situation in order to optimise the sales and profit margin under uncertain influences. Thereby, the most challenging difficulty is the enormous number of alternatives and the complexity of internal relationships, e.g. dependencies between correlating products, and external influences, such as competitor's prices and promotions, market trends, or the general economic situation. A number of approaches have been considered, e.g. regression analysis in economics or systems based on neural networks in computer science [7], but these approaches have significant disadvantages: regress analysis is only able to include a few correlating factors and treats efficiently only linear dependencies, and multi-layer neural networks are in fact able to cope with a much higher number of influencing factors, but they have a fast rise in complexity and correspond to a black box in the

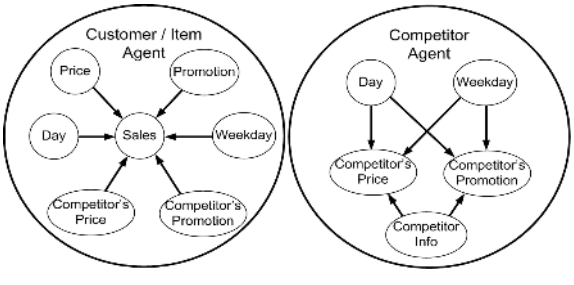


Fig. 12. Simplified SimMarket Agents

sense that they take input and return output without providing any insight on how the individual factors correlate. In contrast, the SimMarket approach tries to understand the individual influences and dependencies between single factors. On this account the approach of SimMarket focuses on representation and simulation of all related objects and entities of the retail domain as individual probabilistic agents. The system includes customer agents which model the shopping behaviour of individual customers, item agents which represent the global sales behaviour of the corresponding item, competitor agents which collect and provide information about competing stores, and other environmental agents modeling the economic situation, important events and weather influences. In order to use these agents for agent-based simulations of certain supermarket stores, their structural fact knowledge bases have to be generated from real data which is stored in data warehouses, i.e. the agents learn sets of individual behaviour networks based on data tables provided by the data bases.

Fig. 12 shows the simplified structure of behaviour networks of customer and item agents as well as of competitor agents.

The network structure of customer and item agents models the sales behaviour of a certain item depending on the item's price and promotion, temporal influences and the competitor's price and promotion activities. The difference between customer and item agents consist of the range of the related data table: the item agent knows the global sales behaviour of the item, whereas the customer agent knows only the customer's individual shopping behaviour concerning the item. The goal of competitor agents is to model the behavior of competitors concerning item price and promotion activities depending on temporal and other related information.

In order to support global simulation of supermarkets, the agents are able to build temporary holons to model dependencies between single agents, e.g. average or addition merging holons of individual customer agents to represent the average or aggregate behaviour of customer groups, relation merging holons of item agents to model dependencies between different items (see Fig. 13) or partial merging holons of customer / item agents and competitor agents to simulate the item sales depending on expected activities of competitors (Fig. 14).

Fig. 13 shows the behaviour net of a holon which results from relation merging of two different item agents. The holon discovered emergent knowledge, i.e. both sales nodes are depending on the other item's price and promotion nodes. Fig. 14 displays

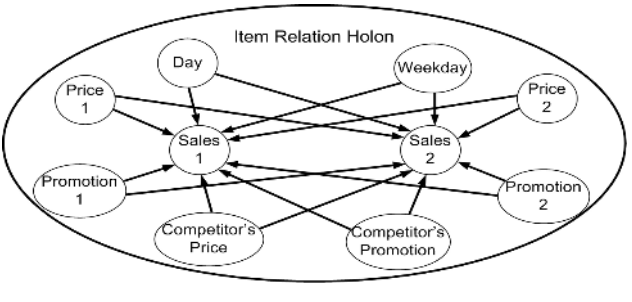


Fig. 13. Item Relation Merging

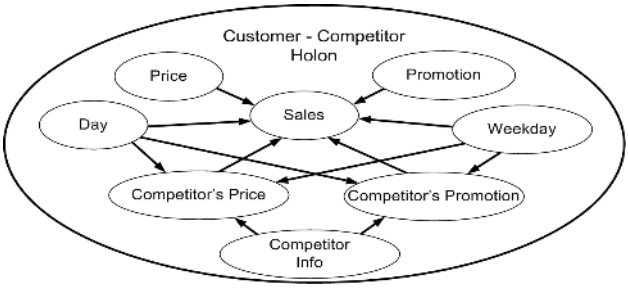


Fig. 14. Customer - Competitor Holon

the result of partial merging of a customer and a competitor agent. The holon is able to simulate the customer's shopping behaviour concerning a certain item depending on the competitor's price and promotion activities which are predicted based on information about the competitor.

Due to the possibility to build temporary holons based on individual agents, the SimMarket simulation environment produces efficiently reasonable results concerning simulation of effects of certain price and promotion changes of single agents within a defined commodity group.

5 Conclusion and Outlook

In this paper we introduced the concept of partial and complete cloning and merging of probabilistic agents based on Bayesian networks in order to increase the performance of holonic multiagent systems. The resulting kinds of holons are useful in agent-based simulation whereas different merging types support different simulation tasks (addition, average and relation merging). In our future work we are developing a measure or a heuristic that can be used to decide during runtime, which kind of holonification seems to be the most efficient for a given problem.

References

1. K. Fischer, M. Schillo, and J. Siekmann, "Holonic Multiagent Systems: The Foundation for the Organization of Multiagent Systems", Proceedings of the First International Conference on Applications of Holonic and Multiagent Systems (HoloMAS'03), 2003
2. G. Weiss, "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence", MIT Press, 1999
3. M. Wooldridge, "An Introduction to Multiagent Systems", John Whiley & Sons, 2002
4. N. R. Jennings, "Agent-based computing: Promise and perils", Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99), 1999
5. A. Koestler, "The Ghost in the Machine", Hutchinson & Co, London, 1967
6. A. Schwaiger, and B. Stahmer, "SimMarket: Multiagent-based Customer Simulation and Decision Support for Category Management", Proceedings of the First German Conference on Multiagent System Technologies (MATES'03), September 2003

7. H. L. Poh, J. Yao, and T. Jasic, "Neural Networks for the Analysis and Forecasting of Advertising and Promotion Impact", Intelligent Systems in Accounting, Finance and Management, Vol. 7, No.4, 1998
8. J. Shen, V. Lesser, and N. Carver, "Minimizing Communication Cost in a Distributed Bayesian Network Using a Decentralized MDP", Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003), 2003
9. N. Friedman, and M. Goldszmidt: Learning *Bayesian Networks with Local Structure*. In: Learning in Graphical Models, Ed. Jordan, M. I., January 16, 2002
10. B. Stahmer and A. Schwaiger, "Holonic Probabilistic Agent Merging Algorithm", Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technologies (IAT), 2004

An Information-Based Agent

John Debenham

Faculty of IT, University of Technology, Sydney, NSW, Australia
debenham@it.uts.edu.au
http://www-staff.it.uts.edu.au/~debenham/

Abstract. An agent architecture is described with high-level process management applications in mind. The agent is "information-based" in that it is designed to operate with real-time information streams. The information that it receives may be of questionable integrity, and is represented in probabilistic first-order logic. The agent applies entropy-based inference techniques to its information base to determine its actions. Each agent is distinguished by the its information and by its plans. The agents' information consists of individual chunks that can be formed into clusters and so into sub-clusters, thus permitting the design of hierarchic multiagent systems consistent with holonic principles.

Keywords: Agent architecture, process management, virtual organisations.

1 Introduction

An "information-based" agent, Π, is described with high-level process management applications in mind. High-level, emergent process management applications generate large amounts of information with varying integrity. Emergent processes are normally supported by CSCW [= Computer Supported Cooperative Work] [1] systems and are not "managed" for two reasons. First, they are not even goal-driven [2] — the processes can "twist and turn" as a process develops. Second, they are driven by the information that they generate, and by — typically large quantities of —background information. Π attempts to fuse the agent interaction with the information that is generated both by and because of it. To achieve this, it draws on ideas from information theory rather than game theory. Π decides what to do — such as whether to contribute to a sub-process — on the basis of information that may be qualified by expressions of degrees of belief. Π uses this information to calculate, and continually re-calculate, probability distributions for that which it does not know. One such distribution, over the set of all possible delegations, expresses Π's belief in the acceptability of a delegated sub-process. The *process delegation problem* belongs to the class of resource allocation games which are inspired by the 'El Farol Bar' problem — see [3] for recent work. Other distributions attempt to predict the behavior of its opponents — such as what delegations they might be prepared to accept. These distribution are calculated from Π's knowledge and beliefs using maximum entropy inference. Π makes no assumptions about the internals of its

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 64–75, 2005.

opponents, including whether they have, or are even aware of the concept of, utility functions. Π is purely concerned with its opponents' behavior — what they do — and not with assumptions about their motivations.

Maximum entropy inference is chosen because it enables inferences to be drawn from incomplete and uncertain information, and because of its encapsulation of common sense reasoning [4]. Unknown probability distributions are inferred using *maximum entropy inference* [5] that is based on random worlds [6]. The maximum entropy probability distribution is "the least biased estimate possible on the given information; i.e. it is maximally noncommittal with regard to missing information" [7]. As applied to process management, maximum entropy inference presents four difficulties. First, it assumes that what the agent knows is "the sum total of the agent's knowledge, it is not a summary of the agent's knowledge, it is all there is" [4]. This assumption is referred to as Watt's Assumption [8]. So if knowledge is absent then it may do strange things. Second, it may only be applied to a consistent set of beliefs — this may mean that valuable information is destroyed by the the belief revision process that copes with the continuous arrival of new information. Third, its knowledge base is expressed in first-order logic. So issues that have unbounded domains — such as time — can only be dealt with either exactly as a large quantity of constants for each possible time step, or approximately as time intervals. This decision will effect the inferences drawn and is referred to as representation dependence [6]. Fourth, maximum entropy can be tricky to calculate — although here the equivalent maximum likelihood problem for the Gibbs distribution was solved numerically without incident by applying the Newton-Raphson method to as many non-linear, simultaneous equations as there are beliefs in the knowledge base. Despite these four difficulties, maximum entropy inference is an elegant formulation of common sense reasoning. Maximum entropy inference is also independent of any structure on the set of all possible deals. So it copes with single-issue and multiple-issue negotiation without modification. It may also be applied to probabilistic belief logic. These properties are particularly useful in managing the multi-issue negotiations involved in managing and coordinating sub-processes. The information-theory oriented analysis described here, which employs maximum entropy inference, is referred to as *ME* in contrast to graph theory *GT*.

Each agent is distinguished by the its information and by its plans. The agents' information consists of individual chunks, represented as statements in first-oder logic qualified with sentence probabilities. So the information can be formed into clusters and then into sub-clusters. In this way, in the context of a process management application, a multiagent system can be designed to reflect the organizational structure — each agent has access to the information that it requires. In this sense, multiagent systems constructed using these "information-based" agents are consistent with holonic principles.

High-level, emergent processes are business processes that are not predefined and are *ad hoc*. These processes typically take place at the higher levels of organisations [9], and are distinct from production workflows [10]. The term

"business process management" [11] is generally used to refer to the simpler class of workflow processes [10], although there are notable exceptions [2]. Emergent processes are opportunistic in nature whereas production workflows are routine [12]. How an emergent process will terminate may not be known until the process is well advanced. The tasks involved in an emergent process are typically not predefined and emerge as the process develops. Those tasks may be carried out by collaborative groups as well as by individuals [13]. Further, the goal of an emergent process instance may mutate as the instance matures. So the goal of an emergent process instance may not be used as a focus for the process management system. From the management perspective, emergent processes are "information-driven". An *information-driven process* is guided by its "process information" and "performance information". *Process information* is information either generated by the individual users or is extracted from the environment, and includes background information. *Performance information* describes how the other agents together with their 'owners' perform, including how reliable they are. Plan-based agent architectures such as BDI [14] are not directly suitable for managing information-driven processes — plans are employed here but they are driven by probability distributions that alter at each time step. In this way the information-based agent is able to "twist and turn" the process management as a process develops.

In this paper, the generic architecture of an "information-based" agent is presented in Sec. 2 — this enables us in Sec. 2.1 to key the work presented here to related work [15] in another application domain. Π's reasoning is described in Sec. 2.2. The essential interactions for managing high-level processes are described in Sec. 3. Sec. 4 discusses the key representation dependence issue in using these information-based agents, and Sec. 5 concludes.

2 The Agent Architecture

An agent called Π is the subject of this discussion. Π engages in multi-issue negotiation with a set of agents: $\{\Omega_1, \cdots, \Omega_o\}$. The foundation for Π's operation is the information that is generated both by and because of its negotiation exchanges. That is, the foundation for Π's thinking is the information in the signals that it receives. Π may not be "utility-aware" — that is, it may not be aware of a utility function. So in the first instance Π draws on ideas from information theory rather than game theory. If Π *does* know its utility function *and* if it aims to optimize its utility *then* Π may apply the principles of game theory to achieve its aim. The objective here is not to reject game theory, but to establish independence from it. In addition to the information derived from the other agents, Π has access to a set of information sources $\{\Theta_1, \cdots, \Theta_t\}$. These information sources may include organizational resources, and general information sources such as email. Together, Π, $\{\Omega_1, \cdots, \Omega_o\}$ and $\{\Theta_1, \cdots, \Theta_t\}$ make up a multiagent system. The integrity of Π's information, including information extracted from general sources, will decay in time [16]. The way in which this decay occurs will depend on the type of information, and on the source

from which it was drawn. Little appears to be known about how the integrity of real information, such as news-feeds, decays, although its validity can often be checked — "Is X head of Department Y?" — by proactive action given a cooperative information source Θ_j.

Π has two languages: $\mathcal{C}$ and $\mathcal{L}$. $\mathcal{C}$ is an illocutionary-based language for communication. $\mathcal{L}$ is a first-order language for internal representation — precisely it is a first-order language with sentence probabilities optionally attached to each sentence representing Π's epistemic belief in the truth of that sentence. Fig. 1 shows a high-level view of how Π operates. Messages expressed in $\mathcal{C}$ from agents in the set $\{\Theta_i\}$ and the set $\{\Omega_i\}$ are received, time-stamped, source-stamped and received in an *in-box* $\mathcal{X}$. The messages in $\mathcal{X}$ are then translated using an *import function* I into sentences in $\mathcal{L}$ that have integrity decay functions (usually of time) attached to each sentence, they are stored in a *repository* $\mathcal{Y}$. And that is all that happens until Π triggers a goal.

Π triggers a goal in two ways: first in response to a message received from an opponent $\{\Omega_i\}$ "I'd like you to join a working group to discuss X", and second in response to some need, $\mathcal{N}$, "goodness, I've got to write the annual report". Π's goals could be short-term such as obtaining some information "when is the annual report due to be completed?", medium-term such as forming a discussion group to determine some issue, or, rather longer-term such as building a (business) relationship with one of the other agents. [The management of business relationships is supported by agent argumentation and is outside this discussion.] For each goal that Π commits to, it has a mechanism for selecting a plan to achieve it. Π's plans reside in a plan library $\mathcal{A}$. Once a plan, a, has been activated, it extracts those sentences from the repository $\mathcal{Y}$ that are relevant to it, instantiates each of those sentences' integrity de! cay functions to the current time t, and selects a consistent sub-set of these sentences using its belief revision[1] function R. Those instantiated sentences that have no decay function are placed into the *knowledge base* $\mathcal{K}^a$, and those that have decay functions are placed along with their sentence probilities into the *belief set* $\mathcal{B}^a_t$. $\mathcal{K}^a \cup \mathcal{B}^a_t = \mathcal{I}^a_t$ is the *information base* created by a at time t. Plan a then derives a set of probability distributions, $\{P_1, \cdots, P_n\}$, from $\mathcal{I}^a_t$ — these probability distributions drive the plan into action. The way in which these derivations are performed are described in Sec. 2.2.

2.1 Π's Languages

The Internal Language $\mathcal{L}$. An *agreement* between two agents Π and Ω is a pair of commitments: $\delta = ((\Pi, \pi), (\Omega, \omega))$ where Π commits to achieve π and Ω commits to achive ω. One predicate in $\mathcal{L}$ is: $\mathrm{Acc_d}(((\Pi, \rho), (\Omega_i, G_i)))$. The proposition $(\mathrm{Acc_d}(\delta) \mid \mathcal{I}_t)$ means: "Π will be comfortable accepting the sub-process delegation agreement δ with agent Ω_i given that Π knows information $\mathcal{I}_t$

[1] This belief revision — consistency checking — exercise is non-trivial, and is not described here. For sake of illustration only, the strategy "discard the old in favor of the new" is sufficient.

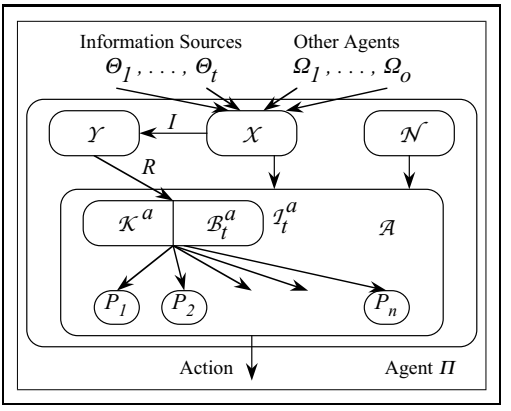


Fig. 1. Basic achitecture of agent Π

at time t". The idea is that Π will accept delegation agreement δ if $\mathbb{P}(\mathrm{Acc_d}(\delta) \mid \mathcal{I}_t) \geq \alpha$ for some threshold constant α. The precise meaning that Π gives to $\mathrm{Acc_d}(\delta)$ is described in Sec. 3. Similarly $\mathrm{Acc_j}$ for Join$(\cdot)$ agreements by which Π commits to join other agents in achieving a cooperative goal. The probability distribution $\mathbb{P}(\mathrm{Agg_d}((\Pi, \rho), (\Omega_i, G_i)))$ is agent Π's estimate of the probability that agent Ω_i will agree to the Delegate agreement δ [or $\mathrm{Agg_j}(\cdot)$ for Join$(\cdot)$ agreements] — it is estimated in Sec. 3.

The architecture illustrated in Fig. 1 has been applied to other application areas besides process management. [15] describes a multi-issue, bilateral bargaining agent Π, with just one opponent Ω, whose strategies are all based on the three distributions: $\mathbb{P}(\mathrm{Acc}(\Pi, \Omega, \delta))$[2] for all deals δ [ie: the probability that Π should accept deal δ from agent Ω], $\mathbb{P}(\mathrm{Acc}(\Omega, \Pi, \delta))$ for all deals δ [ie: Π's estimate of the probability that Ω would accept deal δ from agent Π], and $p_{b,\Omega}$ [ie: the probability of breakdown — the probability that Ω will "walk away" in the next n! egotiation round]. [15] describes how these three complete probability distributions are derived only from information in the signals received by using maximum entropy inference.

The Communication Language $\mathcal{C}$. We assume that the interactions between agents are made within the framework of an infrastructure that fixes ontology and meaning, for instance an Electronic Institution [17]. The communication language $\mathcal{C}$ contains the illocution particle set: $\iota = \{\mathrm{Delegate}, \mathrm{Join}, \mathrm{Accept}, \mathrm{Reject}, \mathrm{Inform}, \mathrm{Quit}\}$ with the following syntax and informal meaning:

- Delegate$((\Pi, \rho), (\Omega_i, G_i))$ means "Π proposes to recompense Ω_i by delivering ρ if Ω_i agrees to take responsibility for an individual goal G_i".
- Join$((\Pi, \rho), (\Omega_i, G_i))$ means "Π proposes to recompense Ω_i by delivering ρ if Ω_i agrees to contribute to cooperative goal G_j".

[2] The predicate Acc$(\cdot)$ is *not* the same predicate as $(\mathrm{Acc_d}(\delta) \mid \mathcal{I}_t)$ used here, although they are similar. We use Acc$(\cdot)$ for consistency with [15].

– Accept(δ) means "the sender accepts your proposed agreement δ".
– Reject(δ) means "the sender rejects your proposed agreement δ".
– Inform($(\Pi, I_k), \Omega_i$) means "Π offers information I_k to Ω_i".
– Quit($\cdot$) means "the sender quits — the interaction ends".

So for these predicates, and in this discussion, an agreement δ has the form $((\Pi, \rho), (\Omega_i, G_i))$. It is commonly accepted since the works by Austin and Searle that illocutionary acts are actions that succeed or fail. We will abuse notation in this paper and will consider that they are predicates in a first order logic meaning 'the action has been performed'. For those more pure-minded an alternative is to consider dynamic logic.

The communication predicates described in the previous paragraph introduce a number of concepts. In the interest of brevity these are only described here informally. The notion of one agent recompensing another [i.e. ρ] refers to both the informal "thanks, I owe you one", and to the formal "take the rest of the day off", or some sum of money. An *individual goal* has the form of: information I_k will be sent to agent Ω_r by time t. An *cooperative goal* has the form of: the assembly of information I_k will be coordinated by agent Π by time t. The expression of the information requires some *ontology* — all of this is not described here.

2.2 Π's Reasoning

Once Π has selected a plan $a \in \mathcal{A}$ it uses maximum entropy inference to derive the $\{P_i\}_{i=1}^n$ [see Fig. 1] and minimum relative entropy inference to update those distributions as new data becomes available. *Entropy*, $\mathbb{H}$, is a measure of uncertainty [5] in a probability distribution for a discrete random variable X: $\mathbb{H}(X) \triangleq -\sum_i p(x_i) \log p(x_i)$ where $p(x_i) = \mathbb{P}(X = x_i)$. Maximum entropy inference is used to derive sentence probabilities for that which is not known by constructing the "maximally noncommittal" [7] probability distribution.

Let $\mathcal{G}$ be the set of all positive ground literals that can be constructed using Π's language $\mathcal{L}$. A *possible world*, v, is a valuation function: $\mathcal{G} \to \{\top, \bot\}$. $\mathcal{V}|\mathcal{K}^a = \{v_i\}$ is the set of all possible worlds that are consistent with Π's knowledge base $\mathcal{K}^a$ — $\mathcal{K}^a$ contains statements which Π believes are true. A *random world* for $\mathcal{K}^a$, $W|\mathcal{K}^a = \{w_i\}$ is a probability distribution over $\mathcal{V}|\mathcal{K}^a = \{v_i\}$, where w_i expresses Π's degree of belief that each of the possible worlds, v_i, is the actual world. The *derived sentence probability* of any $\sigma \in \mathcal{L}$, *with respect to* a random world $W|\mathcal{K}^a$ is:

$$(\forall \sigma \in \mathcal{L}) \mathbb{P}_{\{W|\mathcal{K}^a\}}(\sigma) \triangleq \sum_n \{\, w_n \;:\; \sigma \; is \top \; in \; v_n \,\} \tag{1}$$

The agent's *belief set* $\mathcal{B}_t^a = \{\beta_j\}_{j=1}^M$ contains statements to which Π attaches a *given sentence probability* $\mathbb{B}(.)$. A random world $W|\mathcal{K}^a$ is *consistent* with $\mathcal{B}_t^a$ if: $(\forall \beta \in \mathcal{B}_t^a)(\mathbb{B}(\beta) = \mathbb{P}_{\{W|\mathcal{K}^a\}}(\beta))$. Let $\{\overline{W}|\mathcal{K}^a, \mathcal{B}_t^a\}$ be the "maximum entropy probability distribution over $\mathcal{V}|\mathcal{K}^a$ that is consistent with $\mathcal{B}_t^a$". Given an agent

with $\mathcal{K}^a$ and $\mathcal{B}^a_t$, *maximum entropy inference* states that the *derived sentence probability* for any sentence, $\sigma \in \mathcal{L}$, is:

$$(\forall \sigma \in \mathcal{L})\mathbb{P}(\sigma) \triangleq \mathbb{P}_{\{\overline{W}|\mathcal{K}^a,\mathcal{B}^a_t\}}(\sigma)$$

From Eqn. 1, each belief imposes a linear constraint on the $\{p_i\}$. The maximum entropy distribution: $\arg\max_{\underline{p}} \mathbb{H}(\underline{p})$, $\underline{p} = (p_1, \ldots, p_N)$, subject to $M + 1$ linear constraints:

$$g_j(\underline{p}) = \sum_{i=1}^{N} c_{ji} p_i - B(\beta_j) = 0, \quad j = 1, \ldots, M. \qquad g_0(\underline{p}) = \sum_{i=1}^{N} p_i - 1 = 0$$

where $c_{ji} = 1$ if β_j is $\top$ in v_i and 0 otherwise, and $p_i \geq 0, i = 1, \ldots, N$, is found by introducing Lagrange multipliers, and then obtaining a numerical solution using the multivariate Newton-Raphson method. In the subsequent subsections we'll see how an agent updates the sentence probabilities depending on the type of information used in the update.

Given a prior probability distribution $\underline{q} = (q_i)_{i=1}^n$ and a set of constraints, the *principle of minimum relative entropy* chooses the posterior probability distribution $\underline{p} = (p_i)_{i=1}^n$ that has the least *relative entropy*[3] with respect to $\underline{q}$:

$$\arg\min_{\underline{p}} D(\{p_i\} \parallel \{q_i\}) = \arg\min_{\underline{p}} \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \tag{2}$$

and that satisfies the constraints. This may be found by introducing Lagrange multipliers as above. The principle of minimum relative entropy is a generalization of the principle of maximum entropy. If the prior distribution $\underline{q}$ is uniform, then the relative entropy of $\underline{p}$ with respect to $\underline{q}$ differs from $-\mathbb{H}(\underline{p})$ only by a constant. So the principle of maximum entropy is equivalent to the principle of minimum relative entropy with a uniform prior distribution.

The complexity of the entropy calculations are significantly reduced if the dimensions of the agreement space are ordered and if the agents have clear *preference relations* over them. For example, an agent delegating a sub-process may prefer a shorter delivery time, whereas the agent accepting the delegation may prefer a longer time. The existence of preference relations reduces the number of possible worlds from an exponential function of the domain size to a linear function.

3 The Agent at Work

Π interacts with its opponents $\{\Omega_i\}_{i=1}^n$. It is assumed that goals are initially triggered externally to the system. For example, Π's 'owner' may have an idea that she believes has value, and triggers an emergent process to explore the idea's

[3] Otherwise called *cross entropy* or the *Kullback-Leibler* distance between the two probability distributions.

worth. The interaction protocol is simple, if Π sends a Delegate$(\cdot)$ or a Join$(\cdot)$ message to Ω_i then an interaction has commenced and continues until one agent sends an Accept$(\cdot)$ or a Quit$(\cdot)$ message. This assumes that agents respond in reasonable time which is fair in an essentially cooperative system.

To support the agreement-exchange process, Π has do two different things. First, it must respond to proposals received from Ω_i — that is described below. Second, it must construct proposals, and possibly information, to send to Ω_i — that is described now. Maximum entropy inference is used to 'fill in' missing values with the "maximally noncommittal" probability distribution. To illustrate this suppose that Π proposes to delegate a process to Ω_i. This process involves: Ω_i delivering — using an Inform$(\cdot)$ message — u chapters for a report in so-many days v. This section describes machinery for estimating the probabilities $\mathbb{P}(\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{u,v}))$ where the predicate $\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{u,v}))$ means "Ω_i will accept Π's delegation proposal $((\Pi,\rho),(\Omega_i,G_{u,v}))$".

Π assumes the following two preference relations for Ω_i, and $\mathcal{K}^a$ contains:
$\kappa_{11}: \quad \forall x,y,z((x<y)\rightarrow(\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{y,z}))\rightarrow\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{x,z}))))$
$\kappa_{12}: \quad \forall x,y,z((x<y)\rightarrow(\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{z,x}))\rightarrow\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{z,y}))))$
As noted in Sec. 2.2, these sentences conveniently reduce the number of possible worlds. The two preference relations κ_{11} and κ_{12} induce a partial ordering on the sentence probabilities in the $\mathbb{P}(\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{u,v})))$ array. There are fifty-one possible worlds that are consistent with $\mathcal{K}^a$.

Suppose that Π has the following historical data on similar dealings with Ω_i. Three months ago Ω_i asked for ten days to deliver four chapters. Two months ago Π proposed one day to deliver three chapters and Ω_i refused. One month ago Ω_i asked for eight days to deliver two chapters. $\mathcal{B}^a_t$ contains:
$\beta_{11}:\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{4,10}))$; $\beta_{12}:\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{3,1}))$ and
$\beta_{13}:\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{2,8}))$, and assuming a 10% decay in integrity for each month: $\mathbb{P}(\beta_{11})=0.7$, $\mathbb{P}(\beta_{12})=0.2$ and $\mathbb{P}(\beta_{13})=0.9$.

The distribution $\{\overline{W}\mid\mathcal{K}^a,\mathcal{B}^a_t\}$ has just five different probabilities in it. The probability matrix for the proposition $\mathrm{Agg_d}((\Pi,\rho),(\Omega_i,G_{u,v}))$ is:

$v \diagdown u$	1	2	3	4	5
11	0.9967	0.9607	0.8428	0.7066	0.3533
10	0.9803	0.9476	0.8330	**0.7000**	0.3500
9	0.9533	0.9238	0.8125	0.6828	0.3414
8	0.9262	**0.9000**	0.7920	0.6655	0.3328
7	0.8249	0.8019	0.7074	0.5945	0.2972
6	0.7235	0.7039	0.6228	0.5234	0.2617
5	0.6222	0.6058	0.5383	0.4523	0.2262
4	0.5208	0.5077	0.4537	0.3813	0.1906
3	0.4195	0.4096	0.3691	0.3102	0.1551
2	0.3181	0.3116	0.2846	0.2391	0.1196
1	0.2168	0.2135	**0.2000**	0.1681	0.0840

In this array, the derived sentence probabilities for the three sentences in $\mathcal{B}^a_t$ are shown in bold type; they are exactly their given values. Π's *interaction strategy*

is a function $\mathbf{S} : \mathcal{K}^a \times \mathcal{B}^a_t \rightarrow \mathcal{A}$ where $\mathcal{A}$ is the set of actions that send Delegate(·), Join(·), Accept(·), Reject(·), Inform(·) and Quit(·) messages to Ω_i. If Π sends *any* message to Ω_i then she is giving Ω_i information about herself.

Why would Π accept a Delegate(·) or a Join(·) proposal? Each deal, $\delta = ((\Pi, \rho), (\Omega_i, G_j))$, contains provision for an incentive ρ. However it is more realistic [18] to assume that the agents in an emergent process management system are benevolent [19] — that is, they will accept a responsibility for a process if they believe that they can achieve the process goal. So Π needs machinery to estimate the probability that if it takes responsibility for goal G_j then it will achieve it. The converse problem was considered above: that is, how Π estimates the probability distribution over all possible responses that Ω_i will respond to Delegate(·) or a Join(·) proposals. The proposition $(\text{Acc}_\text{d}((\Pi, \rho), (\Omega_i, G_i)) \mid \mathcal{I}^a_t)$ was introduced in Sec. 2.1. We now describe how the agent estimates $\mathbb{P}(\text{Acc}_\text{d}(\delta) \mid \mathcal{I}^a_t)$ — i.e. the probability that Π attaches to the truth of this proposition for various δ. This is described for delegations only — Join(·) is dealt with similarly.

Π forms its future expectations on the basis of past observations, including the expectations that it has about itself. We described above how Π forms its expectations about its opponent. $\mathbb{P}(\text{Acc}_\text{d}((\Pi, \rho), (\Omega_i, G_i)) \mid \mathcal{I}^a_t)$ is estimated in a similar way — the integrity of past observations is continually discounted, new observations are fed in using minimum relative entropy inference — Eqn. 2. This yields a probability distribution over all possible outcomes that could occur if Π were to commit to a Delegate(·) proposal. Π then uses this distribution to decide whether or not to commit on the basis of the simple criterion: $\mathbb{P}(\text{Acc}_\text{d}((\Pi, \rho), (\Omega_i, G_i)) \mid \mathcal{I}^a_t) > \alpha$ for some personal 'comfort factor' α.

An agent may be motivated to act for various reasons — three are mentioned. First, if there are costs involved in the interaction due *either* to changes in the value of the interaction object with time *or* to the intrinsic cost of conducting the interaction itself. Second, if there is a risk of breakdown caused by a collaborator dropping out of a negotiation. Third, if the agent is concerned with establishing a sense of trust [20] with the collaborator — this could be the case in the establishment of a business relationship. Of these three reasons the last two are addressed here. The risk of breakdown may be reduced, and a sense of trust may be established, if the agent appears to its collaborator to be "approaching the interaction in an even-handed manner". One dimension of "appearing to be even-handed" is to be equitable with the value of information given to the collaborator. Various interaction strategies, both with and without breakdown, are described in [15], but they do not address this issue. An interaction strategy is described here that is founded on a principle of "equitable information gain". That is, Π attempts to respond to Ω_i's messages so that Ω_i's expected information gain similar to that which Π has received.

Π models Ω_i by observing her actions, and inferring beliefs about her future actions in probability distributions such as $\mathbb{P}(\text{Agg}_\text{d})$. Π measures the value of information that it receives from Ω_i by the change in the entropy of this distribution as a result of representing that information in $\mathbb{P}(\text{Agg}_\text{d})$. $\mathcal{I}_t$ is Π's information base at time t. Suppose that Π then receives a message μ giving

$\mathcal{I}_{t+1}$ at time $t+1$. Then the *information* in μ with respect to the information base $\mathcal{I}_t$ is:

$$\mathbb{I}(\mu \mid \mathcal{I}_t) = \mathbb{H}(\mathrm{Agg_d}(\mathcal{I}_t)) - \mathbb{H}(\mathrm{Agg_d}(\mathcal{I}_{t+1}))$$

where the argument of $\mathrm{Agg_d}(\cdot)$ denotes the state of the information base from which it was derived, and $\mathbb{H}(\mathrm{Agg_d}(\cdot))$ is the entropy of the underlying probability distribution from which $\mathrm{Agg_d}$ is aggregated. An alternative way of measuring the value of information is as the Kullback-Leibler distance between the prior and the posterior distributions. If $(p_i)_{i=1}^n$ is the underlying probability distribution from which $\mathrm{Agg_d}(\mathcal{I}_t))$ was aggregated and $(q_i)_{i=1}^n$ the distribution for $\mathrm{Agg_d}(\mathcal{I}_{t+1}))$ then:

$$\mathbb{I}(\mu \mid \mathcal{I}_t) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

More generally, Π measures the value of information received in a message, μ, by the change in the entropy in its entire representation, $\mathcal{J}_t = \mathcal{K}_t \cup \mathcal{B}_t$, as a result of the receipt of that message; this is denoted by: $\Delta_\mu|\mathcal{J}_t^{\Pi}|$, where $|\mathcal{J}_t^{\Pi}|$ denotes the value (as negative entropy) of Π's information in $\mathcal{J}$ at time t. Although both Π and Ω_i will build their models of each other using the same data — the messages exchanged — the observed information gain will depend on the way in which each agent has represented this information. It is "not unreasonable to suggest" that these two representations should be similar. To support its attempts to achieve "equitable information gain" Π assumes that Ω_i's reasoning apparatus mirrors its own, and so is able to estimate the change in Ω_i's entropy as a result of sending a message μ to Ω_i: $\Delta_\mu|\mathcal{J}_t^{\Omega}|$. Suppose that Π receives a message $\mu = \mathrm{Delegate}(\cdot)$ from Ω_i and observes an information gain of $\Delta_\mu|\mathcal{J}_t^{\Pi}|$. Suppose that Π wishes to reject this agreement by sending a counter-proposal, $\mathrm{Delegate}(\cdot)$, that will give Ω_i expected "equitable information gain". This is achieved by:

$$\delta = \{\arg\max_{\delta} \mathbb{P}(\mathrm{Acc_d}(\delta) \mid \mathcal{I}_t) \geq \alpha \mid (\Delta_{\mathrm{Delegate}(\delta)}|\mathcal{J}_t^{\Omega}| \approx \Delta_\mu|\mathcal{J}_t^{\Pi}|)\}$$

That is Π chooses the most acceptable agreement to herself that gives her collaborator expected "equitable information gain" provided that there is such an agreement. If there is not then Π chooses the best available compromise

$$\delta = \{\arg\max_{\delta}(\Delta_{\mathrm{Delegate}(\delta)}|\mathcal{J}_t^{\Omega}|) \mid \mathbb{P}(\mathrm{Acc_d}(\delta) \mid \mathcal{I}_t) \geq \alpha\}$$

provided there is such an agreement — this strategy is rather generous, it rates information gain ahead of personal acceptability. If there is not then Π quits.

4 Representation Dependence

ME is criticized [6] because the way in which the knowledge is represented in $\mathcal{K}^a$ and $\mathcal{B}_t^a$ determines the values derived. This property is promoted here as a strength of the method because the correct formulation of the knowledge base, using the rich expressive power of first-order probabilistic logic, encapsulates features of the application at a fine level of detail.

Time is a common issue in process management applications. Two ways of representing time in logic are: to establish a logical constant for each possible time step, and to work instead with time intervals. Admitting the possibility of an interval containing just one value, the second generalizes the first. To represent time using time intervals, we have to specify the "width" of each interval. Suppose in a delegation that a task will take at least one day. Suppose the predicate $\text{Agg}_{\text{d}}^{\min}((\Pi,\rho),(\Omega_i,G_j),t)$ means "t is the soonest delivery time that agent Ω_i is prepared accept in the delegation of goal G_j from agent Π". This predicate will satisfy: $\forall xy((\text{Agg}_{\text{d}}^{\min}(\delta,x)\wedge\text{Agg}_{\text{d}}^{\min}(\delta,y)\rightarrow(x=y))$. A crude representation of the set of possible time intervals is as two logical constants in $\mathcal{L}$: $[1,2)$ and $[2,\infty)$ where the units are days. Following the development in Sec. 2.2, there are two positive ground literals in $\mathcal{G}$: $\text{Agg}_{\text{d}}^{\min}(\delta,[1,2))$ and $\text{Agg}_{\text{d}}^{\min}(\delta,[2,\infty))$, and there are three possible worlds: $\{(\bot,\bot),(\top,\bot),(\bot,\top)\}$. In the absence of any further information, the maximum entropy distribution is uniform, and, for example, the probability that Ω_i's soonest acceptable delivery time ≥ 2 is $\frac{1}{3}$. Now if the set of possible times had been represented as *three* logical constants: $[1,1.5)$, $[1.5,2)$ and $[2,\infty]$, then the same probability is $\frac{1}{4}$. Which is correct: $\frac{1}{3}$ or $\frac{1}{4}$? That depends on Π's beliefs about Ω_i. In both of these examples, by using *ME*, and by specifying no further knowledge about $\text{Agg}_{\text{d}}^{\min}(\cdot)$, we have implicitly asserted that the probability of each possible world being the true world is the same. In the first example all three are $\frac{1}{3}$, and in the second all four are $\frac{1}{4}$. This is what happens when the "maximally noncommittal" distribution is chosen. Conversely, if believe that: $\forall x,y(\mathbb{P}(\text{Agg}_{\text{d}}^{\min}(\delta,x))=\mathbb{P}(\text{Agg}_{\text{d}}^{\min}(\delta,y)))$ then it is not necessary to include this in $\mathcal{K}^a$. Sec. 1 mentioned Watt's Assumption, that assumption says more than it might at first appear to.

Following from the previous paragraph with just two logical constants, suppose the predicate $\text{Agg}_{\text{d}}^{\text{time}}(\delta,t)$ means "Ω_i is prepared to accept the delegation G_j with delivery time t". Assuming that Ω_i will prefer more time to less, this predicate will satisfy: $\kappa_1:\forall x,y((\text{Agg}_{\text{d}}^{\text{time}}(\delta,x)\wedge(x\leq y))\rightarrow\text{Agg}_{\text{d}}^{\text{time}}(\delta,y))$, where x and y are delivery times and "$\leq$" means "is earlier than". With just κ_1 in $\mathcal{K}^a$ there are three possible worlds: $\{(\bot,\bot),(\top,\bot),(\top,\top)\}$. The maximum entropy distribution is uniform, and, $\mathbb{P}(\text{Agg}_{\text{d}}^{\text{time}}(\delta,[1,2)))=\frac{1}{3}$, and $\mathbb{P}(\text{Agg}_{\text{d}}^{\text{time}}(\delta,[2,\infty)))=\frac{2}{3}$. With no additional information, the distribution $\mathbb{P}(\text{Agg}_{\text{d}}^{\min}(\delta,x))$ will be uniform and $\mathbb{P}(\text{Agg}_{\text{d}}^{\text{time}}(\delta,x))$ will be linear increasing in x.

5 Conclusion

An "information-based" agent has been described for high-level, emergent process management. This agent is based on ideas from information theory. It forms a view on the probability that other agents will respond in certain ways only on the basis of information received — it makes not assumptions about their utility functions. The agent uses entropy based inference that employs a first-order representation language. This means that continuous variables must be represented as intervals. When an issue is represented using intervals there is no "right" or "wrong" choice of intervals. However, choosing the intervals so that

the expected probability distribution of at least one key predicate is uniform over those intervals may simplify the agent's information base.

References

1. Hinds, P.J., Kiesler, S.: Distributed Work. MIT Press, Cambridge, MA (2002)
2. Jennings, N., Faratin, P., Norman, T., OBrien, P., Odgers, B.: Autonomous agents for business process management. Int. Journal of Applied Artificial Intelligence 142 (2000) 145 189
3. Galstyan, A., Kolar, S., Lerman, K.: Resource allocation games with changing resource capacities. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems AAMAS-03. (2003) 145 152
4. Paris, J.: Common sense and maximum entropy. Synthese 117 (1999) 75 93
5. MacKay, D.: Information Theory, Inference and Learning Algorithms. Cambridge University Press (2003)
6. Halpern, J.: Reasoning about Uncertainty. MIT Press (2003)
7. Jaynes, E.: Probability Theory The Logic of Science. Cambridge University Press (2003)
8. Jaeger, M.: Representation independence of nonmonotonic inference relations. In: Proceedings of KR96, Morgan Kaufmann (1996) 461472
9. Jain, A., Aparicio, M., Singh, M.: Agents for process coherence in virtual enterprises. Communications of the ACM 42 (1999) 62 69
10. Fischer, L.: The Workflow Handbook 2003. Future Strategies Inc. (2003)
11. Singh, M.: Business Process Management: A Killer Ap for Agents? In Jennings, N., Sierra, C., Sonenberg, L., Tambe, M., eds.: Proceedings Third International Conference on Autonomous Agents and Multi Agent Systems AAMAS-2004, ACM (2004) 26 27
12. Dourish, P.: Using metalevel techniques in a flexible toolkit for CSCWapplications. ACM Transactions on Computer-Human Interaction (TOCHI) 5 (1998) 109 155
13. Smith, H., Fingar, P.: Business Process Management (BPM): The Third Wave. Meghan-Kiffer Press (2003)
14. Wooldridge, M.: Multiagent Systems. Wiley (2002)
15. Debenham, J.: Bargaining with information. In Jennings, N., Sierra, C., Sonenberg, L., Tambe, M., eds.: Proceedings Third International Conference on Autonomous Agents and Multi Agent Systems AAMAS-2004, ACM (2004) 664 671
16. Bernhardt, D., Miao, J.: Informed trading when information becomes stale. The Journal of Finance LIX (2004)
17. Arcos, J.L., Esteva, M., Noriega, P., Rodrguez, J.A., Sierra, C.: Environment engineering for multiagent systems. Journal on Engineering Applications of Artificial Intelligence 18 (2005)
18. van der Aalst, W., van Hee, K.: Workflow Management: Models, Methods, and Systems. The MIT Press (2002)
19. Huhns, M., Singh, M.: Managing heterogeneous transaction workflows with cooperating agents. In Jennings, N., Wooldridge, M., eds.: Agent Technology: Foundations, Applications and Markets. Springer-Verlag: Berlin, Germany (1998) 219 239
20. Ramchurn, S., Jennings, N., Sierra, C., Godo, L.: A computational trust model for multi-agent interactions based on confidence and reputation. In: Proceedings 5th Int. Workshop on Deception, Fraud and Trust in Agent Societies. (2003)

Designing Communication Protocols for Holonic Control Devices Using Elementary Nets

James Brusey and Duncan McFarlane*

Institute for Manufacturing, Cambridge University,
Cambridge UK CB21RX

Abstract. A difficulty encountered when developing Holonic Manufacturing Systems (HMSs) is the need to place some or all of the intelligence associated with a holon on a small, low-powered device that has real-time constraints. This requirement must be balanced with the need to communicate with such a holon using standard, open, and sophisticated protocols. A possible solution is to split the functionality into two parts with a lightweight sub-holon on the device and an associated high-level sub-holon on a remote server. However a key difficulty is ensuring that the communication protocol between the two sub-holons is robust and reliable, while still allowing all the flexibility that is required in these types of applications. This paper explores the use of Elementary Net Systems as a basis for the specification of the communication and demonstrates that this provides a simple and robust basis for designing correct communication protocols in the case where communication is via a shared data table, such as the memory on an RFID tag.

1 Introduction

The growing use of ICT (Information and Communication Technology) networks in manufacturing systems is encouraging more sophisticated and flexible approaches to measuring and controlling the manufacturing process. It does this by enabling the linking of high-level systems, such as Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM), to low-level, real-time control devices such as Programmable Logic Controllers (PLCs), milling machines, barcode readers, and Radio Frequency Identification (RFID) readers and tags. Stronger links between the high-level systems and the control of the shop floor potentially allows much more flexibility and reconfigurability. Manufacturing operations can be much more responsive to changes in consumer demand and to disruptions in supply from external or internal suppliers.

By linking these systems together, the supervisory controller for a physical device can effectively be distributed across several different computational devices. Following the holonic paradigm [12], the supervisory controller can still be considered to be a single holon associated with a physical device, but internally it is comprised of several sub-holons, one for each *computational* device involved in the control of the physical device.

There are several important reasons for performing this distribution. First, low-level computational devices, such as RFID tags and, to a lesser extent, PLCs, lack sufficient

* The authors are members of the I*PROMS Network of Excellence.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 76–86, 2005.

computational power and memory to be able to communicate with other holons using sophisticated protocols. For example, it is difficult to support high-grade encryption or authentication using such a device, and this might be necessary if insecure communication links are used. Second, the functionality associated with the physical device is divided into that which needs access to sensors and actuators, and possibly has hard real-time requirements, and that which needs to interact with non-real-time entities such as other holons, graphical user interfaces, or databases. The hard real-time requirements are more naturally handled by a specialised device, such as a PLC, while interfacing to GUIs or databases is more easily done on a general purpose computer. Third, it is often useful to maintain state information about a physical device even when communication with that device is intermittent. For example, consider an RFID tag combined with a sensor that is used to monitor the physical state of a component, such as its temperature. It might also be possible for the tag to start a fan if the temperature is too high. When the tag is out of communication range, it might still be desirable to check its last temperature or even to adjust the temperature threshold up or down. This adjustment could then be passed on to the tag when it is next within communication range of a reader.

This paper focuses on the problem of establishing robust and reliable communication between the high and low-level sub-holons within a device holon. This problem is particularly challenging since it

- involves real and non-real-time components,
- must allow for peer to peer communication,
- must be flexible, and allow for reconfiguration,
- must be efficient, and,
- must only require minimal infrastructure, such as that potentially available on an RFID tag.

A basis for performing this communication is a shared data table [3]. However, the data table approach has the difficulty that it is hard to ensure that any particular protocol for passing data is without flaw. A common flaw is for the protocol to produce a *race condition*, a situation where the outcome changes depending on the global sequence of events. For example, consider the case of two holons, both performing the sequence of reading a value from the table (a), then adding one to it and writing it back (b). Given that the holons act independently, we may get the sequence $a_1a_2b_1b_2$ on one occasion and $a_1b_1a_2b_2$ on another. The former causes the value to be incremented by one, while the latter causes it to be incremented by two. In some cases, the variation in behaviour occurs with low probability, resulting in a system which works for days, months or even years before failing. For this reason, testing, however extensive, is not enough to guarantee success.

In this paper, the problem of establishing the correctness of a protocol based on a shared data table is addressed. In particular, the focus here is on the situation where there is no hardware support to ensure synchronisation, other than the low-level protocols that ensure that individual reads or writes are performed correctly and atomically. An approach based on a fundamental class of Petri net is developed. This approach is shown to produce safe protocols.

In the following section, relevant literature is discussed, and the theory of elementary net systems is introduced. Also, the relationship between elementary net systems and ordinary Petri nets is described. The third section develops the theory of dividing the functionality of a single elementary net system over two computationally separate devices.

2 Background

A general mechanism for communicating via shared memory is the semaphore [1]. For the purpose of safely communicating between the sub-holons of a device holon, a semaphore may appear to be an appropriate solution. However semaphores typically require a special machine instruction to lock memory or to "compare-and-swap" and are therefore not useful here.

High-level systems, such as software agents that communicate across a local area network or the Internet and may interact with database systems, tend not to be able to complete their tasks with a hard real-time guarantee. Low-level devices, on the other hand, such as those operating machinery or interacting with parts on a fast moving conveyor, are required to respond within hard real-time constraints. As Brennan *et al.* [3] point out, the interface between such high-level and low-level systems requires an architecture that differs from the architectures used for software agents (such as FIPA). Instead, a Holonic Control Device (HCD) architecture is required. Figure 1 shows a simplified schematic of the HCD architecture, as it applies here. In the diagram, the device holon is divided into high and low-level sub-holons. The low-level has an associated data table, which is also accessible from the high-level sub-holon. The low-level is responsible for communications from sensors and to actuators, while the high-level sub-holon handles all other external communication. Note that using a shared data table is only one possibility and Leitão *et al.* [9] show how a messaging architecture can be developed based on IEC 61499 and CORBA. In their work, however, the emphasis is on a client-server approach with the low-level sub-holon acting as a server.

Given a method for allowing sub-holons to interact, it remains to describe their behaviour. A well established theoretical approach is Supervisory Control Theory

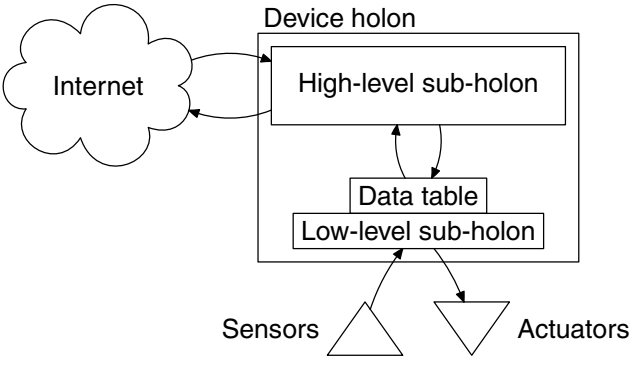


Fig. 1. Simplified schematic of the Holonic Control Device architecture

(SCT) [15,6]. Under SCT, a plant is modelled as a finite state automaton, and then a supervisory automaton is synthesised that controls the behaviour of the plant by inhibiting certain events (assuming that such events can be controlled). Cassandras and Lafortune [6] extend this approach to the use of Petri nets rather than finite state automata. Here, a Petri net supervisory controller is modelled as interacting with the Petri net for the plant. Certain transitions in the plant are controllable, and it is through these transitions that the supervisory controller affects the plant's behaviour. Other work, such as that of Moody and Antsaklis [13], looks at synthesising a supervisory Petri net controller based on a model of the plant and a set of generalised mutual exclusion constraints (GMEC). Such formal methods have provided a strong theoretical basis for developing supervisory controllers with correct behaviours.

Rather than model the plant, Petri nets are also used to directly specify behaviour. For example, Leitão et al. [10] specify holon behaviour using high-level Petri nets. The Sequential Function Chart (SFCs) IEC 61131-3 language has a broad similarity to a Petri net, and so this can also be seen as a way of coding control programs as Petri nets. In addition, several authors, including Brusey et al. [5], Boucher [2] and Frey [8], have described the conversion of Petri nets to Ladder Diagram (LD) code.

Clearly, use of the Petri net formalism is widespread and well accepted in the manufacturing control literature as a basis for synthesising and specifying control programs. In the following section a fundamental class of Petri net, known as an Elementary Net (EN) system, is introduced. On the assumption that the high-level and low-level subholons can be specified in this form, section 3 will show how to test the safety of their interaction.

2.1 Elementary Net Systems

An Elementary Net (EN) system [16] is a fundamental form of Petri net. In comparison with Condition / Event systems (C/E systems) [16], it does not allow reversible transitions, but also does not require every configuration to be reachable. In comparison with Place / Transition systems (P/T systems) [16], EN systems only allow a single token per place, whereas P/T systems allow any number. EN systems have particular properties which can be explained as follows.

An elementary net system is a 4-tuple, $\mathcal{N} = (P,T,F,C_{\text{in}})$, where (P,T,F) is the underlying network consisting of a set of places or states P, a set of transitions or events T and a set of relations or directed arcs F. Transitions are distinct from places $P \cap T = \varnothing$ and directed arcs join either a place to a transition or a transition to a place $F \subseteq (P \times T) \cup (T \times P)$. It is usual to represent the net diagrammatically with circles for places, boxes for transitions, and lines with arrows for directed arcs as shown in figure 2. A *configuration* $C \subseteq P$ corresponds to the dynamic state of the net and is initially C_{in}. The configuration C is represented graphically by the presence of a token (i.e. a large dot) in places that are in C. Roughly speaking, a place is *marked* if it is in C or *unmarked* otherwise. For each $x \in P \cup T$, ${}^\bullet x = \{y \in P \cup T : (y,x) \in F\}$ is the pre-set or set of elements with arcs leading into x, while $x^\bullet = \{y \in P \cup T : (x,y) \in F\}$ is the post-set or set of elements with arcs leading from x. That is for any node x, be it a place or transition, its pre-set ${}^\bullet x$ contains all nodes that have arcs directed to it, while its post-set $x^\bullet$ contains all nodes that have arcs directed away from it. For a set of nodes

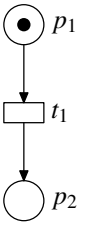


Fig. 2. A simple EN system corresponding to $\mathcal{N} = (\{p_1, p_2\}, \{t_1\}, \{(p_1, t_1), (t_1, p_2)\}, \{p_1\})$

$X \subseteq P \cup T$, its pre-set ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ is the union of pre-sets of its elements and similarly its post-set $X^\bullet = \bigcup_{x \in X} x^\bullet$ is the union of post-sets of its elements.

The state of the net changes by "playing the token game". This game has a simple set of rules [16, page 32], which is restated here.

Lemma 1. *The configuration C of an EN system $\mathcal{N}$ changes according to three rules:*

1. *A transition t is* enabled *to fire at configuration C, denoted $C[t\rangle_{\mathcal{N}}$ (or simply $C[t\rangle$ where there is no ambiguity), if all of its pre-conditions are marked ${}^\bullet t \subseteq C$ and all of its post-conditions are unmarked $t^\bullet \cap C = \varnothing$.*
2. *When a transition t fires at C, tokens are removed from all pre-conditions and added to all post-conditions, yielding a new configuration $C' = (C \cup t^\bullet) - {}^\bullet t$. That C' is the result of t firing at C is denoted as $C[t\rangle_{\mathcal{N}} C'$ (or simply $C[t\rangle C'$).*
3. *A set of transitions $U \subseteq T$ is a step enabled at C, denoted $C[U\rangle_{\mathcal{N}}$ (or simply $C[U\rangle$), iff (a) all transitions in U are enabled at C, and (b) for all $t_1, t_2 \in T$ with $t_1 \neq t_2$, ${}^\bullet t_1 \cap {}^\bullet t_2 = \varnothing$ and $t_1^\bullet \cap t_2^\bullet = \varnothing$.*

The latter condition of the third rule ensures that the firing of any individual transition in U does not affect whether any other transition in U is enabled. Note that, in an EN system, if some preconditions are also postconditions, or in other words, if for any transition $t \in T$, ${}^\bullet t \cap t^\bullet \neq \varnothing$, then the transition t can never be enabled. EN systems that have no such transitions are referred to as *pure*.

Relationship with Ordinary Nets. Elementary nets are less capable models than ordinary Petri nets as they cannot describe a buffer with unbounded capacity. However this is an advantage when used to describe a supervisor intended to be executed by a digital device. Indeed, the binary nature of the state of places means that elementary nets can be easily implemented as computer programs operating on binary memory areas.

As Murata points out [14], it is easy to translate any elementary net into an ordinary net by duplicating each place, reversing its associated arcs, and giving it the opposite initial marking. Similarly buffers larger than one can be dealt with in an elementary net by having a series of places arranged in sequence.

3 P-Communicating Controllers

In this section, we introduce and define the concept of partitioning a monolithic controller into two communicating controllers. A partitioning of the controller is referred

to here as a *P-communicating* controller when the two resulting controllers do not share any transitions, but instead share some places. In other words, they communicate with each other through some subset of their places. The point of partitioning a net is to distribute the computation. Several reasons why computation may need to be distributed were given in the introductory section.

Definition 1. *Let $M = (P,T,F,C_{\text{in}})$ be an elementary net system. A P-communicating partitioning of M means dividing M into two smaller nets*

$$M_i = (P_i, T_i, F \cap (P_i \times T_i \cup T_i \times P_i), C_{\text{in}} \cap P_i),$$

for $i = 1,2$, where the transitions T have been partitioned into disjoint sets T_1 and T_2 so that $T_1 \cap T_2 = \varnothing$, while the places P have been partitioned into non-disjoint sets P_1 and P_2 so that $P_1 \cap P_2 \neq \varnothing$.

Conceptually, the partitioning divides the operations into high-level T_1 and low level T_2 transitions, while dividing the dynamic state into associated purely high-level $P_1 \setminus P_2$, purely low-level $P_2 \setminus P_1$, and shared $P_1 \cap P_2$ places. An example of partitioning a net is shown in figure 3. In the diagram a dashed line is drawn that divides the net into two parts. All transitions are on one side of the partition or the other. Some of the places are on the dividing line and belong to both partitions, while others belong to one partition or the other.

To reduce unnecessary communication, places that are only connected to either high-level or low-level transitions (but not both) can be restricted to either the high or low level, respectively. Therefore the partitioning of P must only include a place in P_i if it is in either the pre-set or the post-set of T_i. Thus

$$P_i = {}^\bullet T_i \cup T_i^\bullet, \tag{1}$$

for $i = 1,2$.

A critical question is whether the partitioning of the monolithic controller into two parts is "safe". A safe partitioning is one that does not change the behaviour. A partitioning is defined as safe in a *particular configuration* if it does not change the *immediate*

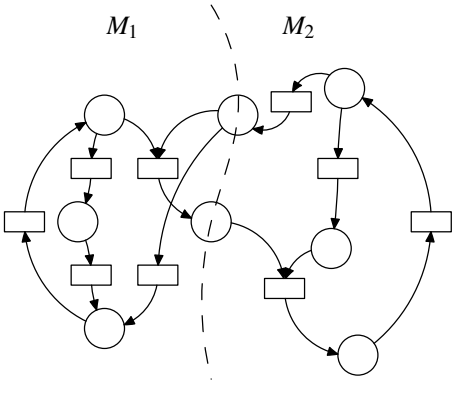


Fig. 3. An example of a P-communicating partitioning of an elementary net

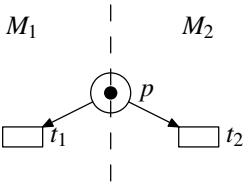


Fig. 4. Conflict between two distributed transitions

behaviour, and this implies that the set of steps enabled at that configuration should be the same. This can be stated formally as follows.

Definition 2. *Let $M = (P,T,F,C_{\text{in}})$ be an EN system. A P-communicating partitioning $M = M_1 \cup M_2$ is safe at some configuration C iff for all steps $U_1 \subseteq T_1, U_2 \subseteq T_2$ where U_1 is enabled in M_1 and U_2 is enabled in M_2, the combined step $U_1 \cup U_2$ is enabled in M. This can be restated as,*

$$(C \cap P_1)\,[U_1\rangle_{M_1} \wedge (C \cap P_2)\,[U_2\rangle_{M_2} \Rightarrow C\,[U_1 \cup U_2\rangle_M\,,$$

for all $U_1 \subseteq T_1, U_2 \subseteq T_2$.

A example of unsafe partitioning is shown in figure 4. The net M is partitioned into M_1, M_2 so that they have respective transitions $T_1 = \{t_1\}, T_2 = \{t_2\}$. Setting $C = \{p\}$ leads to $\{p\}\,[\{t_1\}\rangle_{M_1}$ and $\{p\}\,[\{t_2\}\rangle_{M_2}$ but it is not the case that $\{p\}\,[\{t_1,t_2\}\rangle_M$, so the partitioning is unsafe at $\{p\}$. Note however that for some EN systems, it may be impossible for them to reach a configuration where the partition is unsafe. For example, if we had specified $C_{\text{in}} = \varnothing$, then $C = \{p\}$ would be unreachable. To take this into account, we extend our definition slightly.

Definition 3. *Let $M = (P,T,F,C_{\text{in}})$ be an EN system. A P-communicating partitioning $M = M_1 \cup M_2$ is safe iff for all reachable configurations C, and for all steps $U_1 \subseteq T_1, U_2 \subseteq T_2$ where $(C \cap P_1)\,[U_1\rangle_{M_1}$ and $(C \cap P_2)\,[U_2\rangle_{M_2}$ that the combined step is enabled $C\,[U_1 \cup U_2\rangle_M$.*

However this leaves us with the problem of calculating the reachable configurations first, prior to being able to check whether the partitioning is safe. Although reachability appears to be easily reducible to searching the configuration graph, in fact, it turns out to be NP-complete [7]. Therefore, it would be preferable to have a mechanism to decide whether a partitioning is safe purely on the basis of the structure of the net. It turns out that there is a simple mechanism that can be stated as follows.

Theorem 1. *Let $M = (P,T,F,C_{\text{in}})$ be an elementary net system that has been partitioned into P-communicating M_1, M_2. If transitions from the two partitions do not share preconditions or postconditions, or*

$$^\bullet T_1 \cap {}^\bullet T_2 = T_1^\bullet \cap T_2^\bullet = \varnothing, \tag{2}$$

then the partitioning is safe.

Proof. Choose some steps U_1, and U_2 and some configuration C so that both steps are enabled in their respective partitions. Assume that (2) holds. From lemma 1, the pre-conditions of U_i must hold, so ${}^\bullet U_i \subseteq C \cap P_i$ for $i = 1,2$. Since ${}^\bullet U_i \subseteq P_i$ due to (1), it must be the case that

$$ {}^\bullet (U_1 \cup U_2) \subseteq C. \tag{3} $$

Similarly, post-conditions of U_i must not hold, or $U_i^\bullet \cap C \cap P_i = \varnothing$ for $i = 1,2$, and so

$$ (U_1 \cup U_2)^\bullet \cap C = \varnothing. \tag{4} $$

Finally, for two transitions to be enabled in U_i, they must not conflict, and so,

$$ (\forall t_1, t_2 \in U_i),\ t_1 \neq t_2 \Rightarrow {}^\bullet t_1 \cap {}^\bullet t_2 = t_1^\bullet \cap t_2^\bullet = \varnothing, \tag{5} $$

for $i = 1,2$. Combining with (2), we get

$$ (\forall t_1, t_2 \in (U_1 \cup U_2)),\ t_1 \neq t_2 \Rightarrow {}^\bullet t_1 \cap {}^\bullet t_2 = t_1^\bullet \cap t_2^\bullet = \varnothing, \tag{6} $$

Combining (3), (4), (6) with lemma 1 gives $C[U_1 \cup U_2\rangle_M$ and therefore the partition is safe.

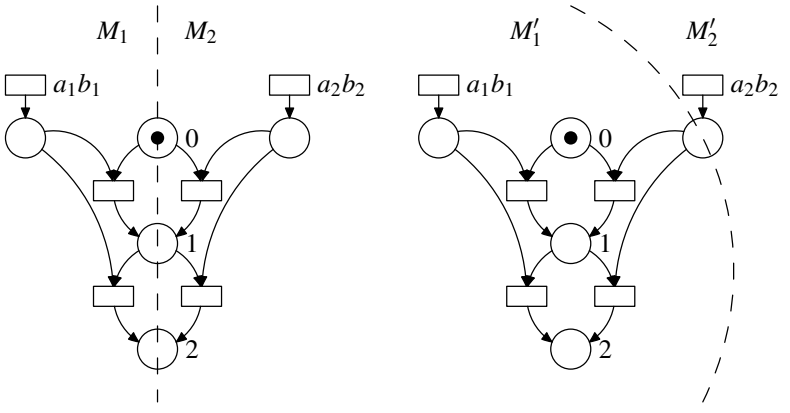
3.1 Additional Partitions

Given the above approach for two partitions, it seems reasonable to ask if it can be extended to N partitions for $N \geq 2$. In the case where a single place is shared amongst $N > 2$ partitions, it is immediate that at least two partitions must have arcs to the place in the same direction. Therefore between these two partitions, (2) does not hold and thus the partitioning may not be safe. On the other hand, if a single place is shared between two partitions at most, and if (2) holds between any two partitions, then the partitioning is safe.

3.2 Example

In this section, a simple example is provided based on the one given in the introduction. For brevity, we show the operation of reading the old value (a), and then incrementing the value and writing it back (b) as a single transition. Note that there is still the potential for a race condition when both $a_1 b_1$ and $a_2 b_2$ happen simultaneously. An EN system corresponding to two processes both incrementing a shared value is shown in figure 5(a). This partitioning is unsafe since there exist transitions on both sides of the partition that have the communicating places $0,1$ as pre-conditions. In addition, transitions on both sides have $1,2$ as post-conditions. The problem is that if both $a_1 b_1$ and $a_2 b_2$ occur simultaneously, this may lead to them both seeing a token in 0, and both moving the token to 1. This is equivalent to the sequence $a_1 a_2 b_1 b_2$. We can modify the partitioning to the form given in figure 5(b) and this makes the partitioning safe. In essence, M_2' now passes a message to M_1' saying "perform the sequence ab on my behalf".

It is useful to think about why the update to the shared place is now safe. If M_2' examines the shared place and finds it to be marked, it cannot fire $a_2 b_2$ and thus cannot affect the shared place's state. Similarly if M_1' examines the shared place and finds it to



(a) Unsafe partitioning (b) Safe partitioning

Fig. 5. Two processes incrementing a value, expressed as an EN system. In (a), the partitioning M_1, M_2 is unsafe, whereas (b) shows a safe partitioning M_1', M_2'.

be unmarked, it cannot fire a transition that will affect the shared place's state. In this way, only one or the other updates the shared place at any time.

3.3 Validation

The general approach used here has been tested in a laboratory environment to control switching gates and docking stations for shuttles on a monorail conveyor. This forms part of a larger holonic environment involving the use of RFID tagged parts [4]. The P-communicating nets are partitioned so that one part resides as ladder logic on a PLC, while the other runs as a Visual Basic program on a separate computer. The Visual Basic program is able to read and write to a shared data table in the PLC. It also acts as an agent, interfacing with the rest of the system, including the RFID readers. A typical interaction would be for the VB agent to be asked to release the shuttle from the docking station. Assuming that the agent is otherwise idle, this would cause it to mark the shared place that had the meaning RELEASE. The PLC would then react to RELEASE being marked by unmarking RELEASE, waiting for the shuttle to arrive, and then releasing it. It would finally mark a shared place indicating that releasing had been DONE. The VB agent reacts to DONE being marked by unmarking it and sending a message to the original requester.

Although this seems quite straightforward, it is interesting that it replaced a system that was also quite simple and usually quite reliable but had a subtle bug. The previous system used a flag to request a release to be performed, equivalent to the RELEASE place, but turned on and off by the VB agent. In some rare cases, the VB agent did not turn off the flag prior to the next shuttle arriving, and thus released two shuttles when only one should have been released. The rarity of occurrence meant that this bug was not discovered for some time. This example is a clear illustration of the importance of formal methods in the design of these types of systems.

4 Conclusions

This paper has presented a method of ensuring safe communication between two or more holons that use a shared data table as the medium.

It may seem surprising, but a simple test proves safe communication without requiring execution testing, and without having to evaluate the dynamic behaviour. In addition, the resulting communication code is extremely lightweight: no code libraries are required, and it can be implemented in a small number of binary operations.

In the case where communication is found not to be safe, a basis has been established for adjusting the partition between the two holons to make it safe. It is planned to address this issue in more detail in future work.

As a final comment, we note that it is possible to augment transitions with additional instructions to be executed when they fire. For example, one transition might access a database, while another might perform a single iteration of A^* search. Of course, the time duration of such operations should be restricted so that the holon remains responsive to its environment and other holons that communicate with it. This approach is comparable to that of the JACK™ agent programming language, which executes individual Java statements as annotations on states of a finite state machine.

Acknowledgements. The authors wish to thank Martyn Fletcher for suggesting the topic, and Mark Harrison for providing useful input on a draft version of this paper.

References

1. Jean Bacon and Tim Harris. *Operating Systems: Concurrent and Distributed Software Design*. Addison Wesley, 2003.
2. Thomas O. Boucher. *Computer Automation in Manufacturing: An introduction*. Chapman & Hall, 1996.
3. R.W. Brennan, K. Hall, V. Mařík, F. Maturana, and D.H. Norrie. A real-time interface for holonic control devices. In Mařík et al. [11], pages 25–34.
4. James Brusey, Martyn Fletcher, Mark Harrison, Alan Thorne, Steve Hodges, and Duncan McFarlane. Auto-ID based control demonstration - phase 2: Pick and place packing with holonic control. Technical report, Auto-ID Centre, Cambridge University, 2003.
5. James Brusey, Martyn Fletcher, Duncan McFarlane, and Alan Thorne. A Petri net compiler for programmable logic controllers. *To appear in IEEE Trans. on Automatic Control*, 2005.
6. Christos G. Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
7. Javier Esparza. Reachability in live and safe free-choice Petri nets is NP-complete. *Theoretical Computer Science*, 198(1–2):211–224, 1998.
8. G. Frey. Automatic implementation of Petri net based control algorithms on PLC. In *Proc. 2000 American Control Conference*, volume 4, pages 2819–2823, Chicago, IL, 28–30 June 2000.
9. Paulo Leitão, Raymond Boissier, Francisco Casis, and Francisco Restivo. Integration of automation resources in holonic manufacturing applications. In Mařík et al. [11], pages 35–46.
10. Paulo Leitão, Armando W. Colombo, and Francisco Restivo. An approach to the formal specification of holonic control systems. In Mařík et al. [11], pages 59–70.

11. Vladimír Mařík, Duncan McFarlane, and Paul Valckenaers, editors. *Holonic and Multi-Agent Systems for Manufacturing: Proc. First Intl. Conf. Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS 2003)*, LNAI 2744, Prague, Czech Republic, September 2003. Springer.
12. D.C. McFarlane and S. Bussmann. Holonic manufacturing control: Rationales, developments and open issues. In S.M. Deen, editor, *Agent-Based Manufacturing*, chapter 13, pages 303–326. Springer-Verlag, Berlin, 2003.
13. J.O. Moody and P. J. Antsaklis. *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Publishers, 1998.
14. Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
15. P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.
16. Wolfgang Reisig and Grzegorz Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*. Springer Verlag, 1998.

A Proposal of Multi-agent Negotiation Mechanism Based on Dynamic Market Concept for Pareto Optimal Solution

Toshiya Kaihara and Susumu Fujii

Kobe University, Department of Computer and Systems Engineering,
Faculty of Engineering, Rokko-dai, Nada, Kobe, Japan
{kaihara, fujii}@cs.kobe-u.ac.jp

Abstract. One of advantages for software agent is their ability to serve as proxies for trade and bartering. This has led to the analysis and development of negotiation mechanism for general resource allocation problem. We consider agent-mediated Pareto optimal allocation of resources through market mechanism. We formulate virtual market model as a discrete resource allocation problem in dynamic situations, and demonstrate the applicability of the market concept with multi-agent paradigm to this framework. In this paper we clarify the proposed mechanism successfully calculates Pareto optimal solutions for the resource allocation problem by comparing our method with conventional analytic approaches. Additionally we apply the mechanism into dynamic market environment, and analyse the rationality of the emerged market by computer simulation.

1 Introduction

The information economy is the merging of trading of traditional markets, the Internet and autonomous agents to form a new market place where agents serve as proxies for buyers, sellers and intermediaries. Evidence of this cam already be seen on the Internet as web sites. Since market price systems constitute a well-understood class of mechanisms that provide effective decentralisation of decision making with minimal communication overhead, we focus on markets for negotiation mechanism of general resource allocation problems. In Market-Oriented Programming (MOP), that is one of the multi-agent protocols for distributed problem solving, the optimal resource allocation for a set of computational agents is derived by computing general equilibrium of an artificial economy [1][2]. Market mechanism can provide several advantages on resource allocation in enterprise negotiation as follows [3][4][5]:

- Markets are naturally distributed and agents make their own decisions about how to bid based on prices and their own utilities for the goods.
- Communication is limited to the exchange of bids and the process between agents as well as the market mechanism.

We have already proposed Virtual Market (VM) concept that is applicable to general resource allocation problem, such as Supply Chain Management (SCM) [6][7].

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 87 – 98, 2005.

Pareto optimality has been already proved in microeconomics, but it has never been clarified in multi-agent based VM approach in those previous researches [6][7]. Hence, it is quite important for the VM research to prove the Pareto optimality of the finally acquired solution through agent negotiations at the first step. Final goal of our research is to apply the VM approach into general resource allocation problems in complex and dynamic situations, or sometimes imperfect information conditions where conventional microeconomics approach never handle, and to implement the VM approach into general negotiation mechanisms in practical situations.

In this paper we construct a Walrasian type VM, that is a principal market model in microeconomics, and formulate agent behaviours considering dynamic situations. Then we try to confirm Pareto optimality of our market model in static environment by comparing the solutions with conventional analytic approach, such as fixed-point algorithm. Finally we apply the proposed mechanism into dynamic market environment, and analyse the rationality of the emerged virtual market by computer simulation.

2 Dynamic Virtual Market

There exists a market-oriented programming to construct a computational market (i. e. virtual market), which consists of several heterogeneous agents [8]. Agent activities in terms of products required and supplied are defined so as to reduce an agent's decision problem to evaluate the tradeoffs of acquiring different products in the market-oriented programming. These tradeoffs are represented in market prices, which define common scale of value across the various products. The problem for designers of computational markets is to specify the mechanism by which agent interactions determine prices.

Market-oriented programming is the general approach for deriving solutions to distributed resource allocation problems by computing the competitive equilibrium of an artificial economy. It involves an iterative adjustment of prices based on the reactions of an agent in the market. Definitions of VM are based on the general equilibrium concept in perfect competitive market, and that means it satisfies a necessary condition of Walrasian type virtual market. As a consequence, a Pareto optimal solution in the market is obtained by VM if the market is in static environment [7]. In this paper we enhance the VM concept into dynamic situations for practical use, because most resource allocations are occurred in dynamic environment practically. The concept of dynamic VM is shown in Fig.1. In the newly proposed dynamic VM, the numbers of all kinds of agents are dynamically changed according to the market performance.

Consumer agents C and producer agents S send their bids to target goods G, and they send supply / demand functions which represent an agent's willingness to sell / buy resources, respectively. They are defined as the relationship between price and quantity of the trading resource. The bidding mechanism computes an equilibrium price in each separate market. It involves an iterative adjustment of prices based on reactions of agents in the market. Agent S submits the supply and demand functions and the auction adjusts individual prices to clear, rather than adjusting the entire price vector by some increment. The mechanism associates an auction with each distinct resource (G). Agents act in the market by submitting bids to auctions. In this paper

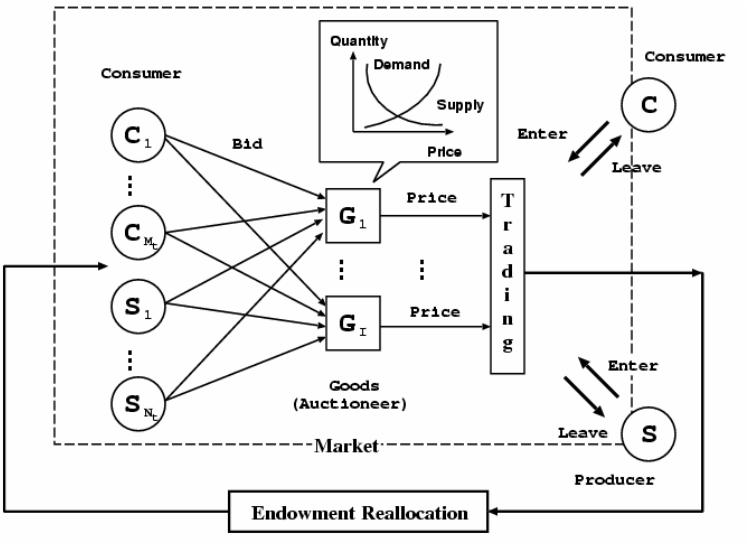


Fig. 1. Dynamic virtual market

bids specify a correspondence between prices and quantities of the resource that the agent offers to demand or supply as a basic study. Given bids from all interested agents, the auction derives a market-clearing price. Each agent maintains an agenda of bid tasks, specifying in which it must update its bid or compute a new one. The bidding process is highly distributed, in that each agent need communicate directly only with the auctions for the resources of interest. Each of these interaction concerns only a single resource; the auctions never coordinate with each other. Agents need not negotiate directly with other agents, nor even know of each other's existence.

As new bids are received at the auctions, the previously computed clearing price becomes obsolete. Periodically, each auction computes a new clearing price if any new or updated bids have been received, and posts it on the tote board. When a price is updated, this may invalidate some of an agent's outstanding bids, since these were computed under the assumption that price for remaining resources were fixed at previous value. On finding out about a price change, an agent arguments its task agenda to include the potentially affected bids. At all times, the market-oriented mechanism maintains a vector of going prices and quantities that would be exchanged at those prices. While the agents have nonempty bid agendas or the auctions new bids, some or all resources may be in disequilibrium. When all auctions clear and all agendas are exhausted, however, the economy is in competitive equilibrium [7][8].

Once the market is converged into an equilibrium situation, then the market organisation is transformed into more activated conditions, i.e. well-balanced market in terms of supply and demand. Consumer / producer agents who notice the market is profitable try to come into and stay it, otherwise they withdraw from it to seek other effective trading markets. Thus the number of agents in the market varies periodically. As a consequence, rational convergence of the market clearing trade is executed in the dynamic conditions based on the supply and demand power balance.

3 Agent Definitions

We describe consumers (i.e. demanders) as c_m (m = 1,2,…,M), and producers (i.e. supplier) as s_n (n = 1,2,…,N). The number of kind of goods g is assumed as I in our dynamic Walrasian VM.

3.1 Demand Agent (Consumer Agent)

3.1.1 Demand Utility

Suppose demand agent c_m has utility function u^{c_m}, which is described with in equation (1). In this equation $x_i^{c_m}$ represents the demand quantity for resource i:

$$u^{c_m} = a^{c_m} \prod_{i=1}^{I} (x_i^{c_m})^{b_i^{c_m}} \tag{1}$$

where $$\sum_{i=1}^{I} b_i^{c_m} = 1 \qquad (0 < a^{c_m}, b_i^{c_m})$$

In this paper we adopt Cobb-Douglas function [2] as a demand function described in equation (1), because the Cobb-Douglas function is one of the primitive functions in microeconomics, which handles economical scale in the market by index constant b.

3.1.2 Budget

Budget of demand agent c_m is formulated by initial quantity of resource (i) : $e_i^{c_m}$, and their price: p_i as follows:

$$B^{c_m} = \sum_{i=1}^{I} p_i e_i^{c_m} + r^{c_m} \tag{2}$$

In this equation r^{c_m} represents supplier's profit, which suppliers return to demanders under zero -profit conditions in the general equilibrium theory.

3.1.3 Bidding Functions

Demand agents send their bid to their target resources in the market, and the bid is formulated as demand function. The function is obtained as the optimal solution in maximising problem of equation (1) under the constraints described in (2). The following demand function is calculated by Lagrange's method of (indeterminate) multiplier in this research.

$$x_i^{c_m}(p_i) = \frac{b_i^{c_m} B^{c_m}}{p_i} \qquad (i = 1, \ldots, I) \tag{3}$$

We assume demander c_m supplies all the initial resources into the market according to the principle of microeconomics, and demander's supply function of resource j is defined as follows:

$$y_j^{c_m} = e_j^{c_m} \tag{4}$$

3.1.4 Agent Entry and Withdrawal

The number of consumer agents changes as the market situation varies. The algorithm of agent's entry and withdrawal is shown in Fig. 2.

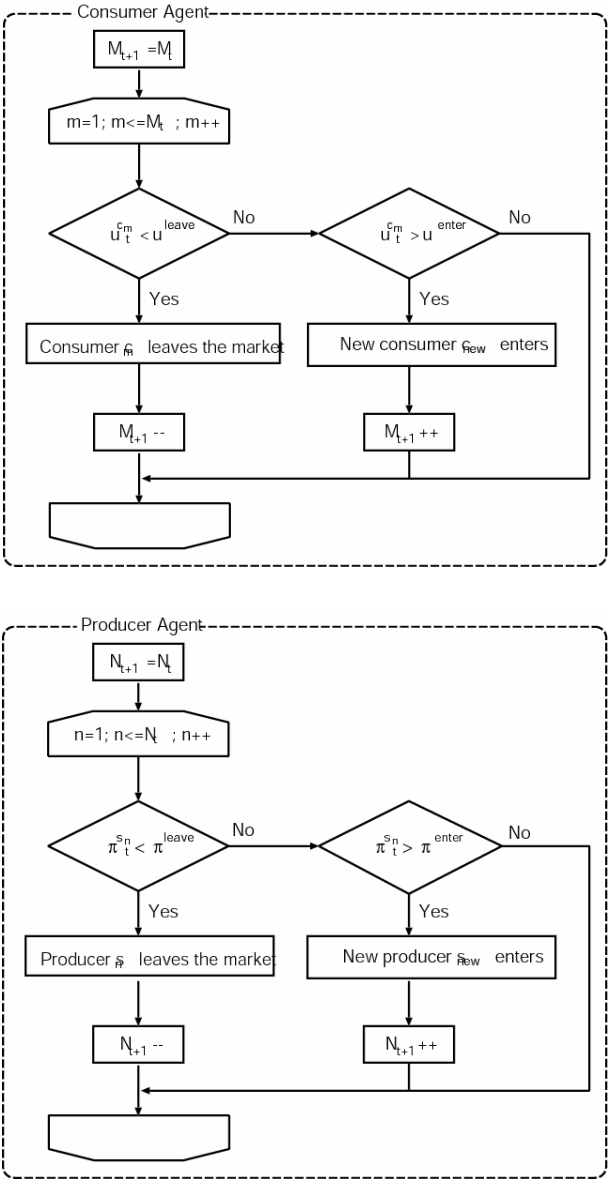


Fig. 2. Agent entry and withdrawal

The number of the entering consumer agents is equivalent of the number of the existing consumer agents whose utility u_i is beyond the pre-set constant value u^{enter}. The parameters of newly entering consumer agent are defined as follows:

$$u^{c_{new}} = a^{c_{new}} \prod_{i=1}^{I} (x_{i,t+1}^{c_{new}})^{b_i^{c_{new}}}, \quad a^{c_{new}} = 1.0$$

$$b_i^{c_{new}} = b_i^{c_m} \times random \quad , \quad e_{i,t+1}^{c_{new}} = random \tag{5}$$

The agents whose utility u_i is below the pre-set constant value u^{leave} leave the market.

3.2 Supply Agent (Producer Agent)

3.2.1 Production Function

As described in demand agent definitions, Cobb-Douglas function is basic functions which handles economical scale in the market easily. In microeconomics production function is assumed to be concave function, and that means market prices are established at a predictable level in the general equilibrium theory in concave shape production function.

We also formulate production function of supply agent s_n to resource j as Cobb-Douglas function to satisfy the assumption, shown as equation (6). Cobb-Douglas function is defined as a concave function in $0< \beta <1$ in this equation.

$$y_j^{s_n} = \alpha^{s_n} (x_j^{s_n})^{\beta^{s_n}} \qquad (0 < \alpha^{s_n}, 0 < \beta^{s_n} < 1) \tag{6}$$

3.2.2 Bidding Functions

According to microeconomics assumption, supply agents have no initial resources. They can earn their profit π^{s_n} by producing value added resources from purchased resources. The profit function is defined as follows:

$$\pi^{s_n} = p_j y_j^{s_n} - p_i x_i^{s_n} \tag{7}$$

Supply agents send supply functions to production resources, and demand functions to purchase resources, respectively. They maximise their profit by solving maximising problem of equation (7) under the constraint in equation (6). We also solve the problem by Lagrange's method of multiplier in this research, and obtain the following demand function and supply function in equation (8) and (9), respectively:

$$x_i^{s_n}(p_i) = \left(\frac{p_i}{\alpha^{s_n} \beta^{s_n} p_j}\right)^{\frac{1}{\beta^{s_n}-1}} \tag{8}$$

$$y_j^{s_n}(p_j) = \left(\frac{p_i^{\beta^{s_n}}}{\alpha^{s_n} (\beta^{s_n})^{\beta^{s_n}} p_j^{\beta^{s_n}}}\right)^{\frac{1}{\beta^{s_n}-1}} \tag{9}$$

3.2.4 Agent Entry and Withdrawal

The number of producer agent also changes as the market situation varies. The algorithm of agent's entry and withdrawal is shown in Fig. 2.

The number of the entering producer agents is equivalent of the number of the existing producer agents whose profit π is beyond the pre-set constant value π^{enter}. The agents whose profit is below the pre-set constant value π^{leave} leave the market.

4 Analytical Method (Fixed Point Algorithm)

We demonstrate the proposed algorithm successfully calculates Pareto optimal solutions by comparing VM solutions with analytic approaches, named fixed point algorithm, and we will explain it briefly below.

Scarf showed that how to compute an approximate Walras equilibrium and proposed a general algorithm for the calculation of a fixed point of a correspondence [9]. This algorithm, named Scarf's algorithm, has been surprisingly efficient to find a general equilibrium that was guaranteed to converge, though does not permit a gradual improvement in the degree of approximation of the solution. In equilibrium point the desired net trades of the two individuals at the equilibrium price ratio balance and the market excess demand functions for the two goods are zero. Computing general equilibria thus implies finding a set of prices corresponding to zero market excess demands for both goods. Although this procedure may work for simple two-commodity markets, difficulties can easily arise for higher-dimensional cases. There are no guarantees that such procedures will converge because the excess demand functions may have local inflection points, changing in slope, or both. Thus, determining prices corresponding to zero excess demands is usually complex.

The Scarf's algorithm applies a procedure to the problem of computing a fixed point of mapping of the unit simples into itself, a mapping whose existence is established by fixed point theorem. In using Scarf's algorithm to find such a fixed point, the unit simplex is divided into a finite number of smaller simplices, each defined by I vertices that are each associated with a label. Each vertex is labelled with the index number of goods, which has the maximum value in market excess demand. The market excess demand of goods i at the vertex is defined as the next equation.

$$E_i = \left(\sum_{m=1}^{M} x_i^{c_m} + \sum_{n=1}^{N} x_i^{s_n}\right) - \left(\sum_{m=1}^{M} y_i^{c_m} + \sum_{n=1}^{N} y_i^{s_n}\right) \tag{10}$$

The algorithm operates by moving through adjacent simplices, deleting and adding vertices. The starting point and each continuation step in the algorithm are such that, prior to finding an approximate equilibria, the algorithm always considers simplices whose verticles have all but one label present, with one label appearing as a duplicate. The procedure involves deleting a vertex with a duplicate label and replacing it with a new vertex, and it then moves to an adjacent simplex. The replacement procedure is such that no simplex once examined can ever be returned to. Because of the finiteness of the number of simplices, and the inability to terminate the procedure other than at a close approximation to an equilibrium, the procedure is guaranteed to find an approximation to an equilibrium. In the limiting process where the number of simplices considered becomes infinite, this approximation will be exact. A point-to-point mapping corresponds to the case in which the Brower fixed point theorem applies [10].

5 Experimental Results

5.1 Experimental VM Model

An experimental 2-producers & 2-consumers market is assumed so as to validate Pareto-optimality of the proposed VM. 2 kinds of goods, goods 1 and goods 2, are traded in this market. The agent parameters of consumer agent and producer agent are given in Table 1 and Table 2, respectively.

Table 1. Parameters in consumer agent

Agent	Utility function (u^{c_m})	Initial endowment ($e_1^{c_m}, e_2^{c_m}$)	Initial utility
c_1	$1.0(x_1^{c_1})^{0.3}(x_2^{c_1})^{0.7}$	(30.0, 10.0)	13.90
c_2	$1.0(x_1^{c_2})^{0.5}(x_2^{c_2})^{0.5}$	(20.0, 40.0)	28.28

Table 2. Parameters in producer agent

Agent	Input	Output	Production function
s_1	goods 1	goods 2	$y_2^{s_1} = 2.0(x_1^{s_1})^{0.7}$
s_2	goods 1	goods 2	$y_2^{s_2} = 3.0(x_1^{s_2})^{0.5}$

5.2 Experimental Results in Static Environment

We prepared an experimental static VM (Fig. 3) so as to validate Pareto optimality by comparing fixed point algorithm, as basic analysis. Agent entry or withdrawal, described in 3.1.4 and 3.2.4, is not activated in the static model, so the numbers of agents are never changed, and remain 2 for each kind of agent in this experiment. Consumer agents send their bids as supply & demand functions to both the goods homogeneously. Producer agents send their demand & supply function to goods 1 & goods 2, respectively. That means they both produce goods 2 from goods 1 in this market. Simulation results on price and trade changes of goods 1 are shown in Fig. 4.

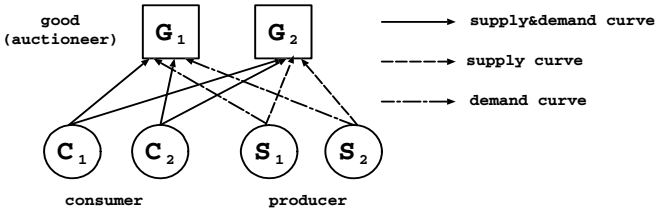


Fig. 3. 2-producer 2-consumer market

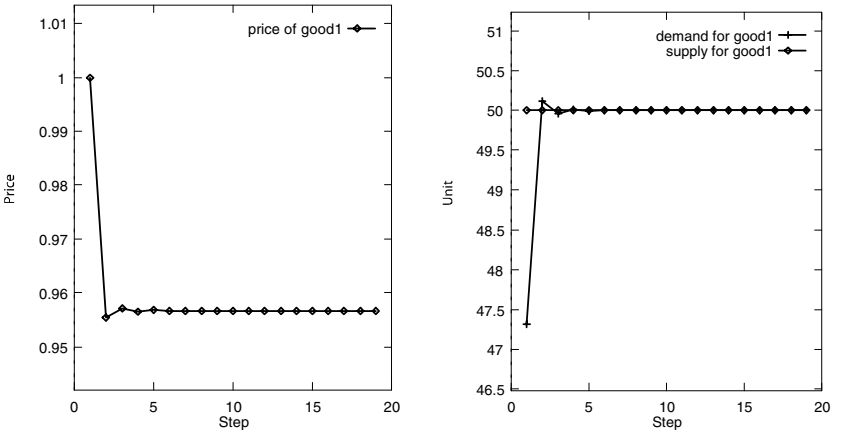


Fig. 4. Price and trade convergence

In this model we obtained equilibrium price vector $\hat{p} = (\hat{p}_1, \hat{p}_2)$ as follows:

Equilibrium price vector $\hat{p} = (0.95673, 0.98543)$

Normalised equilibrium price vector $\hat{p} = (0.49261, 0.50739)$

And final resource allocation in consumer agent and producer agent are shown in table 3 and 4, respectively. It is observed that the total consumer's utility is increased in table 7 compared with table 2, because the producer's profit is returned to consumer agents in this market under the zero-profit conditions in the general equilibrium theory. The utility of agent c_1 is increased by 17.2 % especially. That is because of the agent's preference to goods 2 as well as the zero-profit conditions. In this market all the producer agents supply goods 2, and that increases the utility of agent c_1. On the other hand, the utility of agent c_2 is slightly decreased, because it becomes slightly difficult to get both goods due to the stronger supply flow from producer agents to agent c_1.

Table 3. Resource allocation of consumer agents

Agent	Consumption $(x_1^{c_m}, x_2^{c_m})$	Utility
c_1	(12.67, 28.69)	22.45
c_2	(31.56, 30.64)	31.10

Table 4. Resource allocation of producer agents

Agent	Consumption	Production	Profit
s_1	3.39	4.70	1.39
s_2	2.39	4.63	2.28

We applied fixed point algorithm in this market model to confirm Pareto optimality of the VM solutions. We calculated a fixed point of mapping of the unit simples into itself, a mapping whose existence is established by fixed point theorem. In using Scarf's algorithm to find such a fixed point, the unit simplex is divided into a finite number of smaller simplices (i.e. grid size). In this paper the grid size is set to 100,000 for the precise comparison. We obtained equilibrium price vector $\hat{p} = (\hat{p}_1, \hat{p}_2)$ by the fixed point algorithm as follows:

Equilibrium price vector $\hat{p} = (0.49262, 0.50739)$

It is obvious that the equilibrium price set obtained by Walrasian VM is almost equivalent to the one from the fixed point algorithm, and that means VM solutions have been confirmed to be converged into Pareto optimal. The small difference is caused by the grid size of the fixed point algorithm. We compared the calculation time between VM approach and fixed point algorithm. CPU consumption time (second) for each approach in this market model is as follows:

VM	: 0.0017 (sec.)
Fixed point algorithm	: 12.025 (sec.)

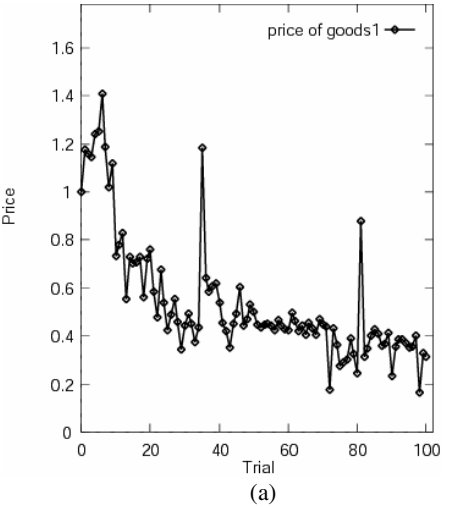
VM approach is obviously more than 7,000 times as fast as the analytic approach in this model. It has also been proved that the proposed VM based approach is much more practical in terms of calculation time to obtain Pareto optimal solutions in resource allocation problems.

5.3 Experimental Results in Dynamic Environment

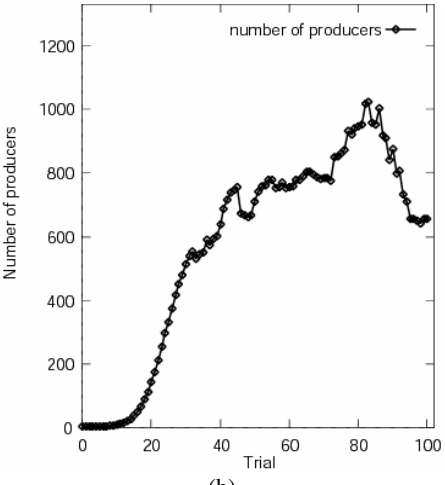
We constructed an experimental dynamic VM, and the parameters defined in 3.1.4 and 3.2.4 are as follows:

$$u^{enter} > 4.5,\ u^{leave} < 4.5,\ \pi^{enter} = 1.0,\ \pi^{leave} = 0.0.$$

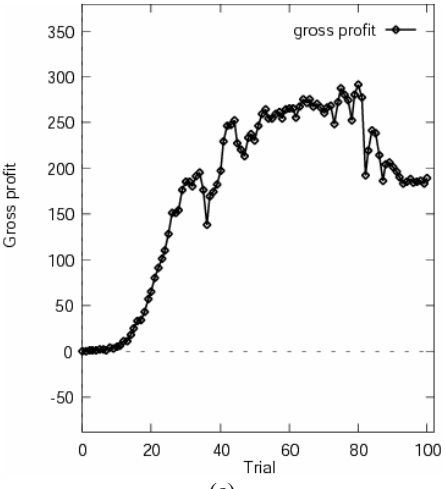
Price fluctuation of goods 1, producer agent utility, and the number of total producer agents under dynamic environment is shown in Fig. 5(a), (b), (c), respectively.



(a)



(b)



(c)

Fig. 5. Experimental results in Dynamic VM

It is obvious that the price is gradually converged into equilibrium conditions even in the dynamic VM in Fig. 5(a). Pareto-optimal solution is obtained at each trial in Fig. 5(a), because the dynamic VM consists of the consecutive static VM. So it is quite interesting to attain the convergence in dynamic VM, because the VM emerged adaptively a macro rational situation through the interaction between micro-agent behaviour and macro market environment with Pareto-optimal situations.

The number of producer agents gradually increased once, and finally converged in Fig. 5(b). Producer agents are permitted to join the market on their decision, and it has been clarified that the appropriate number of producer agents are almost 620 in this experiment. Important fact is the appropriate number was autonomously found in the

dynamic environment. As the number of producer agents increases, the gross profit of producer agents is also enlarged, and that is quite reasonable result.

As a result we have confirmed that the proposed algorithm is quite effective to calculate a Pareto optimal solution in general resource allocation problems by taking a metaphor of economic systems.

6 Conclusions

In this paper we described the formulation of agents in dynamic VM at first. Then we have clarified the Pareto optimality of our market model in static environment by comparing the solutions with conventional analytic approach, such as fixed-point algorithm. We have also shown our algorithm is efficient in calculation time. Finally we applied the proposed mechanism into dynamic market environment, and it has been confirmed that the dynamic VM successfully conducts equilibrium solutions with interacting micro agent behaviour and macro market environment adaptively. As a result we have confirmed that the proposed algorithm is quite effective to calculate a Pareto optimal solution in general resource allocation problems by taking a metaphor of social economic systems.

Our next step is to apply the proposed methodology into practical resource allocation problems in dynamic situations, and clarify the efficiency of our idea based on dynamic VM.

References

1. Shoven J. B. and J. Whalley, Applying General Equilibrium, Cambridge University Press (1992)
2. Kreps, D. M., A Course in Microeconomic Theory. Harvester Wheatsheaf, New York (1990)
3. Gehring, Thomas, Intermediation in Search Markets, Journal of Economics and Manegement Strategy, vol.2 (1993) 97-120
4. Rust, J and Hall, G, Middlemen versus Market Makers: A Theory of Competitive Exchange, Journal of Political Economy, vol. 111, no. 2 (2003) 353-403
5. Gode, D. K. and Sunder, S., Allocative Efficiency of Markets with Zero Intelligence Traders: Market as a Partial Substitute for Individual Rationality. Journal of Political Economy, Volume 101, Number 1, February (1993) 119-137
6. Kaihara T., Supply Chain Management based on Market Mechanism in Virtual Enterprise, Infrastructures for Virtual Enterprises, L. M. Camarinha-Matos and H. Afsarmanesh Eds., Kluwer Academic Publishers, Boston (1999) 399-408
7. Kaihara T., Supply Chain Management with Market Economics, International Journal of Production Economics, Elsevier Science, Vol. 73, Issue 1 (2001) 5-14
8. Wellman M. P., A Market-Oriented Program-ming Environment and its Application to Distributed Multi- commodity Flow Problems, ICMAS-96 (1996) 385-392
9. Scarf, H. E., The computing of Economics Equilibria, New Heaven, Yale University Press (1973)
10. Brouwer, L.E.J., Beweis der Invarianz der Dimensionzahl, Math. Ann. 70, (1911) 161-165

Integrating Transportation Ontologies Using Semantic Web Languages

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. Communication between agents is possible only when agents understand ontologies used in exchanged messages. Since we cannot hope that one universally accepted ontology would be ever created, approaches allowing using different ontologies by different agents must be used. In this paper, we present a way of expressing mappings between ontologies and using these mappings for communication between agents. To illustrate our approach, we have chosen several ontologies that describe transportation domain and created partial mappings between them. The mappings are expressed as matching forward rules which can be applied to messages exchanged between agents. In this way, messages are translated between different ontologies, as we demonstrate on an implementation that uses Jade and Jena packages. Using this approach, agents from different departments within one company or from different companies can be integrated even when they use different ontologies.

1 Introduction

Manufacturing systems, like pure computer systems, are changing from isolated islands to more and more interconnected networks, and are forming multi-agent systems [15]. These systems are useful not only for flexible and robust integration within a shop floor, but are also necessary for integration of various departments within a company, and also for integrating multiple companies to create virtual enterprises [19].

Communication between these agents is possible only when agents understand ontologies used in exchanged messages. As it was shown [13], we cannot hope that one universally accepted unchanging ontology even for a small domain would be ever created, approaches allowing to use different ontologies by different agents for must be used. In this paper, we present a way of expressing mappings between ontologies and using these mappings for communication between agents. To illustrate our approach, we have chosen three ontologies that describe transportation domain and we have created partial mappings between these ontologies. The mappings are expressed as pattern matching forward rules which can be applied to messages exchanged between agents. In this way, messages are translated between different ontologies. The translation service is provided by a specialized ontology agent. We briefly describe an implementation

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 99–110, 2005.

using Jade [11] and Jena [12] packages for multi-agent systems and semantic web applications. Using this approach, agents from different departments within one company or different companies can be integrated even when they use different ontologies.

The rest of this paper is organized as follows. We briefly describe ontologies and semantic web technologies with the focus on the Web Ontology Language OWL [3]. Then we outline the three selected ontologies. After that, we show examples of mapping rules. We show how these rules are used for translation between messages exchanged between agents. We conclude with conclusion and outline of the possible future work.

2 Ontologies

"Ontology" can be defined as an explicit specification of conceptualization [8] of a domain. Ontologies capture the structure of the domain, i.e. conceptualization. This includes model of the domain with possible restrictions. The conceptualization describes knowledge about the domain, not about the particular state of affairs in the domain. In other words, the conceptualization is not changing, or is changing rarely. Ontology is then specification of this conceptualization — the conceptualization is specified by using particular modeling language and particular words. Formal specification is required in order to be able to process ontologies and operate on ontologies automatically.

Ontology is describing the domain, while a knowledge base (based on an ontology) describes particular state of affairs. Each agent has its own knowledge base, and only what can be expressed using an ontology can be stored and used in the knowledge base. When an agent wants to communicate to another agent, he uses the constructs from some ontology. In order to understand in communication, ontologies must be shared between agents.

When agents do not share their ontologies, the problem of semantic interoperability arises. One solution of this problem, especially when existing ontologies used by agents cannot be changed, is translation between ontologies. This problem is being solved not only in multi-agent systems, but also in the area of Semantic Web [23].

2.1 Semantic Web

Semantic Web "provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C [1] with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming." [23]

Many languages were proposed for ontology modeling for the semantic web. These converged to the Web Ontology Language OWL [3]. OWL was designed by

[1] World Wide Web Consortium, `http://w3.org/`.

W3C and is based on the experience gained with the DAML+OIL [2] language. OWL describes domain in the 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 in fact a description logic embedded into RDF triples [14] and uses RDF/RDFS and XML Schema [20] constructs.

2.2 Resource Description Framework RDF

The Resource Description Framework (RDF) [14] serves for describing resources, i.e. web documents identifiable by Uniform Resource Identifiers[2]. It is based on the model of triples object-predicate-subject. All three elements of a triple can be resources. The subject can be in addition a literal (e.g. string, number). These triples together form a graph that serves for the representation of data, information and knowledge. Resource Description Framework Schema (RDFS) provides primitives in a form of resources with a given explicit meaning. It allows stating basic taxonomical constructs and basic constraints on properties.

2.3 Web Ontology Language OWL

The Web Ontology Language, OWL, is built on RDF and RDFS. 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 [3]:

- 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.

OWL divides the universe into two disjoint parts. One part is the datatype domain described by the XML Schema [20] 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`. 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)

[2] `http://www.ietf.org/rfc/rfc2396.txt`

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. Properties can be either object properties that relate objects to other objects, or datatype properties that relate objects to datatype values. Datatype values are defined by XML Schema definitions. Properties can be refined similarly as in the case of classes together with special additional restrictions such as domains, ranges, transitiveness, etc.

The Web Ontology Language OWL has become W3C recommendation [3], and is becoming widely accepted. Software packages are available [12] that enable to process OWL and to reason over OWL ontologies and knowledge bases.

3 Ontologies for Transportation

To illustrate our approach to integrating multi-agent systems, we have chosen an important part of the domain of manufacturing or virtual enterprises — transportation. In this section, we will briefly describe selected ontologies that describe the transportation domain.

3.1 Berlin Transportation Ontology

This ontology [24] has a goal to describe Berlin local transport service, but defines general concepts and relations for general transportation description (see figure 1[3]). This ontology was created within the Agentcities project.

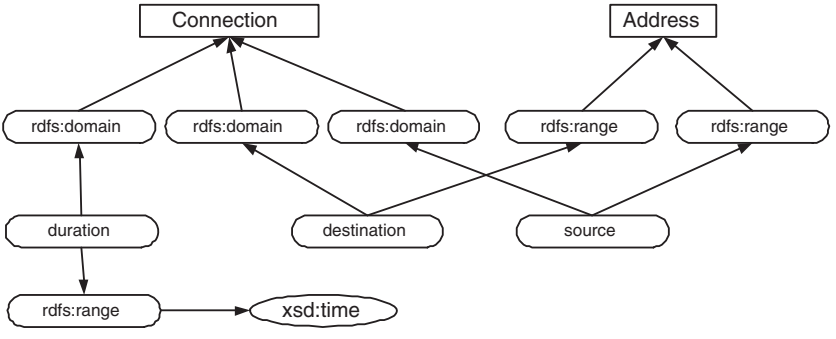


Fig. 1. Berlin transportation ontology (upper part with main concepts)

This ontology is expressed in DAML. The main classes are “Connection” and “Address”, that can be connected via relations “destination” and “source”.

[3] In the figures 1, 2, 3, ontologies are expressed in RDFS/OWL. A rectangle corresponds to a class, a rounded rectangle corresponds to an object property, and an oval corresponds to datatype property.

Based on these main entities, the ontology then describes in more details city means of transport, such as buses or taxis.

We don't know whether this ontology was actually used, but we note that without our modifications, Jena [12] was not able to parse the original ontology. Also, there is one data-type property that has range defined as a subclass of `daml:Class`, so the ontology itself may be viewed as inconsistent. We fixed these problems, and translated the DAML ontology into OWL ontology using a script.

3.2 Boeing Transportation Ontology

We named this ontology [1] as "Boeing" one only because of its author affiliation. We don't know whether this ontology is actually used in Boeing.

This ontology is expressed in the Knowledge Interchange Format KIF [7] and is designed in a layered way. First, a graph ontology describing graph theory with nodes and arcs is described (see figure 2). Using this ontology, a basic transportation ontology is introduced, that uses graph "node" as transportation "place" and graph "arc" as transportation "conduit". The transportation ontology defines above all transporting action and transportee, and defines constraints on them. Then, as an example, a car ontology is introduced, that is based on the transportation ontology. This ontology for example specializes "transporting" to "driving", "vehicle" to "car", and defines additional constraints (such as that "person" is "driving", which is not always the case of the more general concept "transporting").

We translated this ontology from KIF to OWL manually. All concepts and relations were translated to OWL, including basic constraints expressed in KIF. Since KIF is more expressible than OWL, the translation was lossy. Some of the axioms (such as more complicated rules) could not be expressed in OWL.

3.3 Material Transportation Ontology

The material transportation ontology [21] was originally designed for material transportation in a factory. Its use is described e.g. in [21].

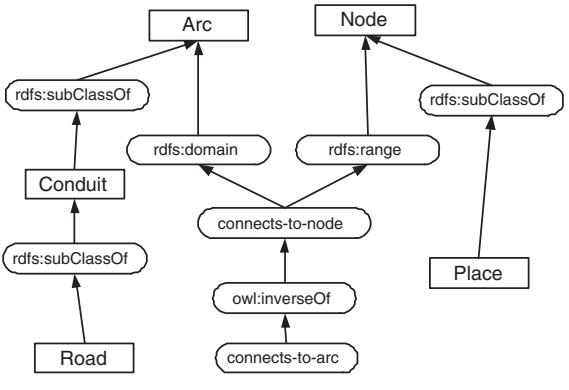


Fig. 2. Boeing transportation ontology (upper part with main concepts)

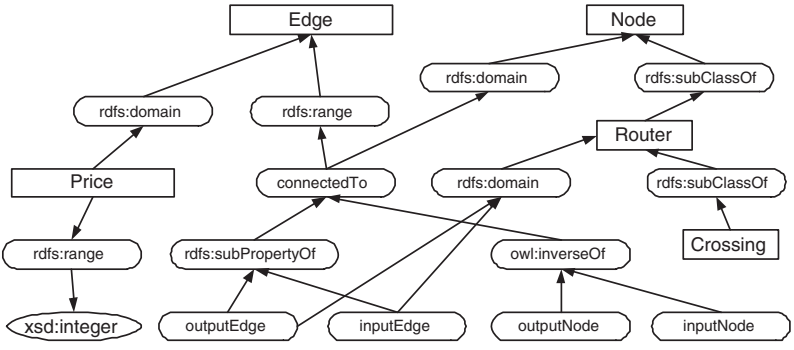


Fig. 3. Material transportation ontology (upper part with main concepts)

This ontology is expressed as the list of XML tags and attributes with their meaning described using natural language in a tabular form (the same approach is used for all ontologies in FIPA [5] specifications). Based on this, we have created a transportation ontology expressed in OIL[4] [4], and added the axioms corresponding to the natural language description. The OIL ontology is described in [16]. For the purposes of this paper, we have converted this OIL ontology into OWL format (see figure 3) automatically.

4 Integrating Ontologies

As we can see from the previous section and figures 1, 2, 3, ontologies describing very similar domains are not exactly the same. Agents from different companies are likely to use different ontologies, and still they should be able to communicate at least on the level where their domains and ontologies overlap. We can observe that at least the top levels of our selected ontologies describe very similar concepts. However the exact conceptualization and its specification is different, since the goal of each ontology is different — each one was designed for different purposes.

To enable agents to communicate, a translation between messages can be used. We propose a way of mapping ontologies using rules that are expressed using semantic web languages. As soon as these mappings are established (i.e. rules are available), they can be used for translation of messages between different ontologies.

Rules are generally viewed as the next layer over ontologies and logic for achieving the goals of the semantic web [23]. Several rules languages built on OWL were proposed. The most known is probably Semantic Web Rule Language SWRL [9]. In this paper, we will use different syntax as used by the Jena [12] rules, however our rules can be translated to SWRL easily. The syntax and semantics of these rules is described in [12].

[4] Ontology Inference Layer — a description logic that is the base for the semantics of DAML+OIL and later OWL.

4.1 Translation Between Transportation Ontologies

Let us illustrate the mapping specification on a few examples. We will start with the mapping from the Boeing transportation ontology to the Material one. The easiest rule is for one to one mapping, where a class or property name can be translated exactly to another class or property name in the other ontology. Of course, this is possible only when such lossless and exact translation exists. In our case, we can translate "Arc" directly to "Edge". The corresponding rule needs to be specified for two cases — when the class is used in subject and when the class is used in object in the triples[5]:

```
[(boeing:Arc ?p ?o) -> remove(0) (material:Edge ?p ?o)]
[(?s ?p boeing:Arc) -> remove(0) (?s ?p material:Edge)]
```

Note that we are specifying one way translation only. In this particular case, "Edge" can be translated back to "Arc", but generally, this is not always the case for any two concepts. For example, "Conduit" from Boeing ontology doesn't have its direct counterpart in the material ontology. So, even when "Conduit" can be translated to "Edge" (with the loss of specialization), "Edge" cannot be translated to "Conduit", but only to "Arc". Translation of "Edge" is illustrated in the first case in the figure 4, translation of "Conduit" is illustrated in the second case in the same figure.

The same approach can be used for one to one mappings of properties. For example, in the mapping from the material transportation ontology to the Berlin one, the following rules can be used:

```
[(?e material:outputNode ?n) -> remove(0)
                                    (?e berlin:destination ?n)]
[(?e material:inputNode ?n) -> remove(0) (?e berlin:source ?n)]
```

The source ontology may have concepts that do not have direct counterparts in the target ontology, but these counterparts can be easily constructed — e.g. by joining more concepts in a union (which is illustrated as the third case in the figure 4) or by adding additional constraint. In both cases, a new concept is in fact created in the target ontology, and is then used for translation. In the case of classes, this new concept would typically correspond to an RDF anonymous node. However, it is not possible to use anonymous node for property definition, so a new property must be explicitly created.

For example, when translating from the Boeing transportation ontology to the Berlin one, then the property "connects-to-node" can be translated to the union of "destination" and "source". The rule then looks as follows[6]:

[5] The triples in the second part of the rule are added when the triples in the first part of the rule are found in the message that should be translated (i.e. if there is a match for the triple pattern). The `remove(i)` primitive removes the i-th matched triple.

[6] In RDF triples, this means that a new property destsrc is created as the union of destination and source properties, i.e. `destsrc owl:unionOf (destination source)`. The `makeTemp` primitive will create a new anonymous RDF node.

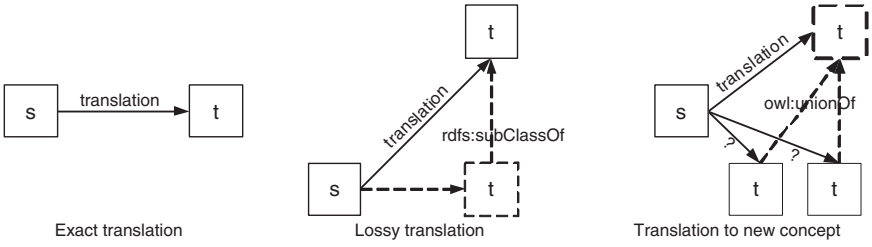


Fig. 4. Illustration of translations from source to target ontologies in various situations

```
[(?e boeing:connects-to-node ?n) makeTemp(?a1) makeTemp(?a2) ->
 remove(0) (?e berlin:destsrc ?n) (berlin:destsrc owl:unionOf ?a1)
 (?a1 rdf:first berlin:destination) (?a1 rdf:rest ?a2)
 (?a2 rdf:first berlin:source) (?a2 rdf:rest rdf:nil) ]
```

Let us add that the rules can use many additional primitives for more complicated matchings or target triples creations. Using the rules as explained above, mappings can be created between different ontologies. These rules are then directly used to translate messages between ontologies.

Note that the head of the rule can contain multiple concepts, and so a rule expressing n:m translation can be defined and used as well. This allows to describe more complex mappings than the ones described in the figure 4. Also, the primitives in the rules include arithmetical operations, so a conversion for example between different metric systems in different ontologies can be expressed and applied.

4.2 Implementation in Multi-agent System

We have implemented an ontology agent that enables other agents to cope with multiple ontologies. This agent is based on our previous work, where we designed and implemented ontology service agent [18], that helps other agents with maintaining ontologies, with querying them and with providing reasoning over them. The idea of this agent is based on the FIPA ontology agent proposal [6], but we modified it to use semantic web technologies — RDF and OWL. The agent is implemented in Jade [11] and its architecture is illustrated in the figure 5.

For translation, we use the forward chaining pattern matching rules (in the Jena form mentioned above) that are directly processed and used for reasoning and translation by the Jena package. The ontology agent maintains ontologies and is now enhanced from [18] to maintain mappings between ontologies expressed in Jena rules, and to translate messages between ontologies on request. For the message flow between ordinary agents and the ontology service agent we refer to our previous work [18].

4.3 Example Translation

To give an overview of the whole process, we will illustrate one example message translation. An agent that uses Boeing transportation ontology is informing

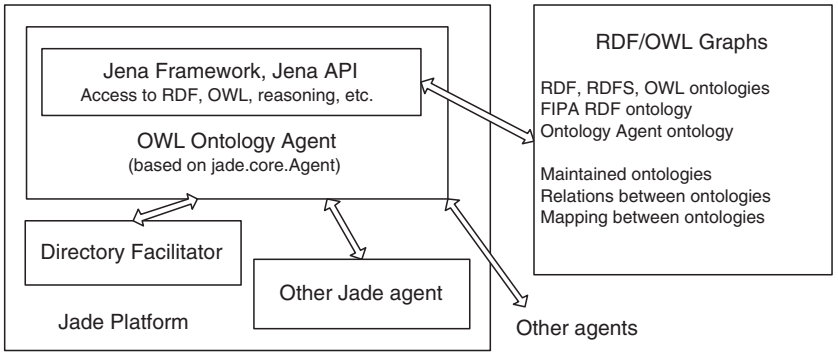


Fig. 5. Architecture of the ontology agent implemented in Jade and accessing knowledge and ontologies expressed in RDF/OWL using Jena

about a connection between two places (for example as an answer to a query, or as otherwise requested information). The message content (for the rest of the FIPA ACL message using OWL, please refer to [18]) is:

```
@prefix : <http://boeing.com/transportation/A1KB#> .
@prefix boeing: <http://boeing.com/transportation/Ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
:placeA a boeing:Place.
:placeB a boeing:Place.
:path1 a boeing:Conduit; boeing:connects-to-node :placeA, :placeB.
```

The rules fired during translation are as follows (note that these are only selected rules fired during this particular translation; the whole mapping from Boeing to Berlin ontology is of course much bigger):

```
[(?e boeing:connects-to-node ?n) makeTemp(?a1) makeTemp(?a2) ->
 remove(0) (?e berlin:destsrc ?n) (berlin:destsrc owl:unionOf ?a1)
 (?a1 rdf:first berlin:destination) (?a1 rdf:rest ?a2)
 (?a2 rdf:first berlin:source) (?a2 rdf:rest rdf:nil) ]
[(?s ?p boeing:Place) -> remove(0) (?s ?p berlin:Address)]
[(?s ?p boeing:Conduit) -> remove(0) (?s ?p berlin:Connection)]
```

The message translated from Boeing transformation ontology to the Berlin one, as returned by the ontology service agent, is then:

```
@prefix : <http://boeing.com/transportation/A1KB#> .
@prefix berlin: <http://berlintransport.com/Ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
:placeB a berlin:Address .
:placeA a berlin:Address .
berlin:destsrc owl:unionOf (berlin:destination berlin:source).
:path1 a berlin:Connection; berlin:destsrc :placeB, :placeA .
```

5 Related Work

There are many proposed ways of integration of information — the state of the art in ontology mapping is surveyed e.g. in [22]. They can be classified into three main approaches:

- relate all information sources to one big and always evolving ontology
- provide individual mappings between individual ontologies
- hybrid approaches

In our work, we provide individual mappings between individual ontologies; however, we note that ontology mappings are needed in all of these three approaches, so our work is relevant to all of them. Unlike other approaches that use purely syntactical rewriting, the semantic web rules used in our work operate directly on the information and ontology level. This ensures semantics expressed in a better way and thus easier maintenance of mappings.

An important issue is finding mappings between ontologies. For the purposes of the work described in this paper, we established the mappings manually. A fully automatic mapping is hardly possible without other information than just unrelated ontologies. With additional information, such as shared instances between ontologies, at least some mappings could be found automatically [17]. Statistical approaches are often used to find similarities; however, they require human interaction. In addition, they introduce uncertainty in the mapping rules, which are acceptable for tasks like internet search, but are not acceptable for manufacturing systems communication, where understanding must be guaranteed. In our approach, the mapping is exact (i.e. there are no confidence measures for the rules). When translation cannot be found, the information is passed in the original ontology and it is on the receiving agent to handle this situation.

Also, the problem of semantic interoperability can be made easier by using common foundational upper level ontologies. A working group was established within IEEE that attempts to develop a standard upper ontology [10], but even this group agreed that foundations for working with multiple ontologies is needed more than a single huge ontology. When a common upper ontology is used, then at least the upper concepts of the derived ontologies can be easily mapped. However, even when finding mapping is in this case easier and more reliable, the need for mapping does not disappear.

6 Conclusion and Future Work

We have presented a way of enabling communication between agents that use different ontologies. Our approach relies on a specialized ontology agent that has

information about mappings between ontologies and is able to use them to help other agents to translate between ontologies. We have illustrated this approach on selected ontologies from transportation domain.

The current implementation requires that the mapping is told to the ontology agent — it has no capabilities to learn these mappings. A method of learning to map between simpler ontologies using shared instances was presented in our previous work [17]. We plan to extend this method to employ constraints available in ontologies to be able to find mappings between ontologies more easily.

The presented work enables to integrate agents when they use different ontologies. This is a necessary requirement for larger scale integration, such as for virtual enterprises, where whole companies of different kinds (and so using different ontologies) are involved in one process.

Acknowledgements

This work is supported by the grant MSM 6840770013 of the Ministry of Education, Youth and Sports of the Czech Republic and the EU Network of Excellence I*PROMS. We would like to thank anonymous referees for their helpful comments which helped us to improve the paper.

References

1. Clark, P.: Transportation Ontology. (1998) http://www.cs.utexas.edu/users/pclark/kr-web/other/passenger-vehicle/transportation.km
2. DAML — Darpa Agent Markup Language Website. http://www.daml.org/
3. Dean, M., Schreiber, G. (eds): OWL Web Ontology Language Reference. W3C Recommendation. (2004) http://www.w3.org/TR/owl-ref/
4. Fensel, D., Horrocks, I., van Harmelen, F., Decker, S., Erdmann, M., Klein, M.: OIL in a Nutshell. Knowledge Acquisition, Modeling and Management. EKAW Conference. Lecture Notes in Computer Science Vol. 1937. Springer-Verlag, Berlin Heidelberg New York (2000) pp 1-6
5. FIPA — Foundation for Intelligent Physical Agents Website. http://www.fipa.org/
6. FIPA: Ontology Service Specification. (2001) http://www.fipa.org/specs/fipa00086/
7. Genesereth, Michael R., Fikes, Richard E.: Knowledge Interchange Format, Version 3.0, Reference Manual. (1994) http://logic.stanford.edu/kif/Hypertext/kif-manual.html
8. Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition. Vol. 5, Issue 2 (1993) pp 199-220
9. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language — Combining OWL and RuleML. (2004) http://www.daml.org/rules/proposal/
10. IEEE P1600.1 Standard Upper Ontology Working Group Home Page. (2005) http://suo.ieee.org/
11. Jade — Java Agent DEvelopment Framework website. (2005) http://sharon.cselt.it/projects/jade/

12. Jena 2 — A Semantic Web Framework website. (2005) http://www.hpl.hp.com/semweb/jena2.htm
13. Madnick, Stuart E.: From VLDB to VMLDB (Very MANY Large Data Bases): Dealing with Large-Scale Semantic Heterogeneity. Proceedings of the 21st VLDB Conference (1995)
14. Manola, F., Miller, E. (eds): RDF Primer. W3C Recommendation. (2004) http://www.w3.org/TR/REC-rdf-syntax
15. Mařík, V., McFarlane, D.: Industrial Adoption of Agent-Based Technologies. IEEE Intelligent Systems. Vol. 20, No. 1 (2005) pp 27-35
16. Obitko, M.: Translating Between Ontologies For Agent Communication. Gerstner Laboratory for Intelligent Decision Making and Control. Series of Research Reports. No. 149/02. (2002) http://cyber.felk.cvut.cz/gerstner/reports/GL149.pdf
17. Obitko, M., Mařík, V.: Mapping between Ontologies for Agent Communication. Multi-Agent Systems and Applications III. Lecture Notes in Computer Science Vol. 2691. Springer-Verlag, Berlin Heidelberg New York (2003) pp 191-203
18. Obitko, M., Mařík, V.: Ontology Service Agent with Web Ontology Language. European Workshop on Multi-Agent Systems (EUMAS) Proceedings. (2004) pp 477-488
19. Pěchouček, M., Vokřínek, J., Bečvář, P.: ExPlanTech: Multiagent Support for Manufacturing Decision Making. IEEE Intelligent Systems. Vol. 20, No. 1 (2005) pp 67-74
20. Thompson, H. S., Beech, D., Maloney, M., Mendelsohn, N.: XML Schema Part 1: Structures. Second Edition. W3C Recommendation. 2004. http://www.w3.org/TR/xmlschema-1/
21. Vrba, P., Hrdonka, V.: Material Handling Problem: FIPA Compliant Agent Implementation. Multi-Agent Systems and Applications II. Lecture Notes in Computer Science Vol. 2322. Springer-Verlag, Berlin Heidelberg New York (2002) pp 268-279
22. Wache, H., Vgele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hbner, S.:Ontology-Based Integration of Information A Survey of Existing Approaches. IJCAI–01 Workshop: Ontologies and Information Sharing (2001) pp 108-117
23. W3C Semantic Web Homepage. http://www.w3.org/2001/sw/
24. Zhang, T.: Local Transport Ontology. (2003) http://www.agentcities.org/EURTD/Ontologies/local-transport.v2.daml

A Strategy to Implement and Validate Industrial Applications of Holonic Systems

Francisco P. Maturana[1], Raymond J. Staron[1], Pavel Tichý[2], Petr Šlechta[2], and Pavel Vrba[2]

[1] Rockwell Automation, 1 Allen-Bradley Drive, Mayfield Hts., OH 44124-6118, USA
{fpmaturana, rjstaron}@ra.rockwell.com
[2] Rockwell Automation s.r.o., Pekarska 10a, 15500 Prague 5, Czech Republic
{ptichy, pslechta, pvrba}@ra.rockwell.com

Abstract. Classical control systems are based on feedback techniques and models that generally cannot manage computational complexity, nonlinearity and uncertainty. Moreover, classical control cannot adapt well to the variability of the processes under control in a dynamic fashion. However, agent-based control eases combinatorial complexity by enabling a robust partitioning of knowledge and behaviors. It is a difficult challenge to create the infrastructure, development system and validation tools for agent systems. In this paper we discuss fundamental steps to achieve the foundation infrastructure for creating agents but also we address several guidelines to create the agents and the requirements to present this to non-agent specialists.

1 Introduction

Agent/Holonic technology provides an appropriate framework to integrate knowledge with efficient production actions in distributed organizations. Integration of knowledge depends on balanced information representation within and across heterogeneous organizations. Integrating information within a specific environment can be helped by the deployment of standards and open systems. However, it is harder to attempt such a smooth integration with the information of multiple heterogeneous subsystems. It is one of the missions of this document to discuss the requirements and steps for building flexible agent systems in a friendly manner. This architecture provides a first step toward deployment and integration of multi-agent systems in distributed automation and control, which is an essential aspect of holonic manufacturing systems [1-5].

Intelligent autonomous control provides the ability to address large complex systems. However, reliance upon a single controller presents a survivability problem, as damage to that centralized controller or to the communications infrastructure used to interact with actuators and sensors, can result in a loss in controllability. This is especially applicable when the system in question operates within a hazardous and response critical environment.

In a real production place there are several rules that need to be satisfied before an automation solution can be considered viable [4]. Since agent architectures are con-

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 111 – 120, 2005.

ceived as counterparts of the physical hardware, the rules affecting hardware and network configuration, responsiveness and reliability, etc. also affect the agents. These rules are hard requirements that necessarily impose practical considerations onto the construction of agent-based control. One endeavor of this work is to show how to extend a commercial architecture to use agents. To provide a truly distributed, survivable and adaptable architecture, it is necessary to create infrastructure for hosting agents and all the necessary mechanism for communication.

2 General Architecture of Agents

We use a distributed control architecture based on automation control devices with extended firmware. With these extensions, component-level intelligence is possible and the physical devices can be converted into intelligent nodes with negotiation capabilities. The intelligence of the system is distributed among multiple controllers by placing standalone or multiple agents inside the controllers. The relationship among the agents is loosely coupled but these are cohesive and adaptable. The agent architecture is organized according to the following characteristics [4] [10]:

- Autonomy: Each agent makes its own decisions and is responsible for carrying out its decisions toward successful completion;
- Cooperation: Agents combine their capabilities and simple rules of interaction into collaboration groups (clusters) to adapt and respond to events and goals;
- Communication: Agents share a common language;
- Fault tolerance: Agents possess the capability to detect equipment failures and to isolate failures from propagating; and
- Pro-action: Agents periodically or asynchronously propose strategies to enhance the system performance or to prevent the system from harmful states.

2.1 Automation Architecture

One primary requirement for agent-based systems is the distribution of intelligence and control. Figure 1 shows the architecture of the extended firmware, which is common to all control products of the Logix family. The Logix products are automation controllers from Rockwell Automation: ControlLogix, CompactLogix, FlexLogix, and SoftLogix controllers. The modified firmware includes the following layers: 1) Distributed Control Agent (DCA) and 2) Intelligent Agents. The DCA layer glues the components using threads. The Intelligent Agents layer represents the agent objects that are downloaded into the controller as user defined programs. The agents interact with control programs or device drivers in a similar manner as the interaction of head and body in a holonic architecture. The agents use a Job Description Language (JDL) and FIPA envelops [2] to encode messages. There are different types of messages such as REQUEST and INFORM.

The agents can contact any node in the network as long as the physical connectivity permits it using a JDL message. A JDL message is written in a common-to-all-agents language. JDL represents planning, commitment, and execution phases during the task negotiation. Information is encoded as a sequence of hierarchical actions with precedence constraints.

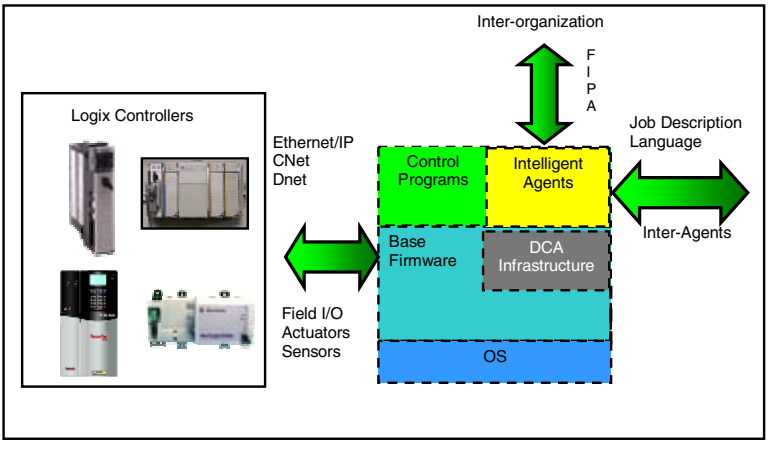


Fig. 1. Automation control device

A second use of JDL is the encoding of plan templates. When an agent accepts a request, an instance of a plan template is created to record values emerging during the planning process. If a part of the request cannot be solved locally, the agent sends a partial request to other agents as a sub-plan. This is possible if there are other agents associated with the missing capability.

The distribution of agent capabilities is a critical design decision. Essentially, the capabilities should be distributed according to the equipment characteristics. Each physical component provides a different set of services at each node. Since the production environment has natural redundant nodes, it makes sense to create a template description of capabilities. Moreover, the capabilities are organized according to the process flow. Nonetheless, the distribution of capabilities is still a design decision.

Agents are not restricted to only infra messages but they can contact external foreign organizations using FIPA messages. FIPA offers the framework to build such messages and we exploit it by making it part of the communication system inside the controllers. The FIPA standard uses a matchmaking mechanism to locate agents. Currently, the architecture includes a full implementation of Directory Facilitators (DF), also called middle agents, that provides matchmaking and Agent Management Services (AMS) functionality.

During the creation, downloading and activation of agents into the controllers, the agents follow a registration procedure with a local DF inside the hosting controller. A local DF is nothing else but a local cache where the agents register their capabilities. The capabilities represent the services an agent offer to the system. The capabilities can be arranged in a hierarchical manner for knowledge and precedence constraint consistency. Although the MAS are naturally distributed, there is still a need to organize knowledge in hierarchies. Another aspect of interest is the study of hierarchies that can emerge dynamically in virtual clusters.

2.2 Agent Architecture

Agents are goal-oriented entities that act autonomously and cooperatively when involved in problem-solving tasks. They organize their individual capabilities around

system goals. A system goal is abstract because there is not explicit declaration of it during the design of the system. A goal can be described as a dynamic social force that puts agents together to solve a particular task. Thus, a system goal has the following attributes: (a) System event or need, (b) A first and second level responders, (c) Plan of actions, and (d) Execution.

The physical system has several processes that generate events either from the device level up to the agent or from the super user (agent or human) down to the agent. There is an event triggering a specific response inside the agent program. Due to the distribution of capabilities there are particular agents associated with the events. The first level responders act accordingly by carrying out local actions and by propagating the events to other agents. This then introduces the second level responders which are contacted via capability matching. The first and second level agents coordinate a possible plan and establish the execution action to respond to the event. An event is not limited to reactive types only. There may be events autonomously triggered by the agent themselves. The agents use their rules to trigger such events. Figure 2 shows the agent architecture. There are four components:

- Planner: This component is the reasoning engine of the agent. It reasons about plans and events emerging from the physical domain or other agents;
- Equipment Model: This component is a decision-making support system. Models of the physical domain are placed here to help the Planner evaluate different configurations. The Equipment Model provides metrics for proposed configurations;
- Execution Control: This component acts as control proxy, which translates committed plans into execution control actions; and
- Diagnostics: This component monitors the health of the physical device. It is programmed with a model of the physical device, where a set of input parameters is evaluated according to a model to validate the readings.

The physical device (e.g., valve) is operated according to a sequence of actions. These actions are encapsulated inside *device drivers* and classified according to the type of OEM. The device driver becomes a template library for the physical device. When an agent is created for such a device, an instance of its device driver is copied in the controller's memory. The sensors and actuators associated with the physical device are connected to the automation controller using standard networks. State variables are contained in the controller's data table (memory).

The agents essentially consist of two parts: (1) ladder logic (fast control) and (2) C++ (but not restricted to) code (high-level control: planning, negotiation, diagnostics). We cannot use reference implementations because of language, memory and speed restrictions.

Currently, the planner engine processes XML scripts (declarative style of programming). However, we found that this approach is not sufficient for all applications, so we are extending the system to a procedural programming. Advantages of procedural programming are: better performance, better flexibility (extensions may be introduced). Disadvantages are that this requires more code for the agent behavior. It may introduce more complexity in guiding the user through the agent design process.

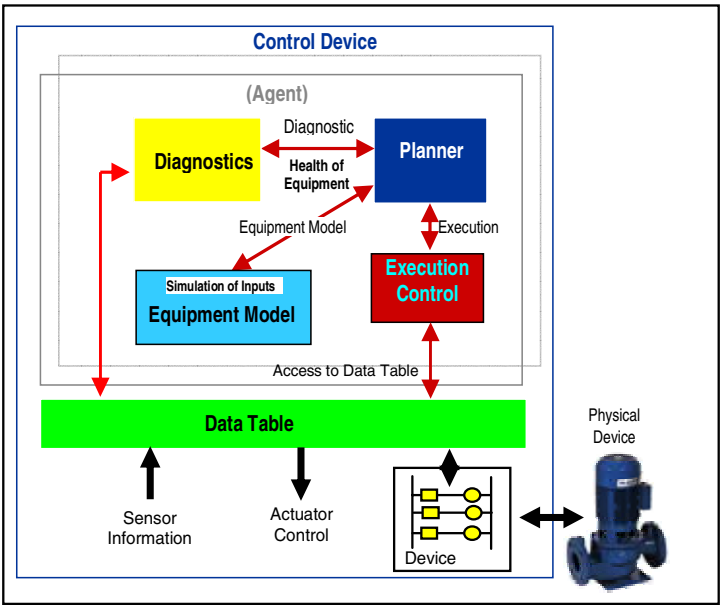


Fig. 2. Agent architecture

As first step, it is essential to provide an infrastructure to contain agents. The implementation of agents should be flexible enough such that the user can design it in a clear and consistent manner. Most of the burden of creating the agents must be encapsulated in the agent design tool. The tools should provide enough flexibility to program any desire behavior for ensuring a complete modeling of the physical environment. To help this process, we present a subset of rules in the following section.

3 Rules to Consider When Modeling Agents

It has been difficult to establish rules of thumbs to create agent-based systems because each problem studied presented different constraints, system properties and design perspectives. We studied a vast range of problems including steel mill factory control, material distribution systems, logistics, water circulation systems, shipboard automation, power, etc. [7-10]. In each case there was a need to establish problem specific rules without being able to extend the same solution to another problem. It was possible to foresee Meta rules to guide the design of the agents and control.

A fundamental decision to make when designing agents is the partitioning of the agent behavior versus control behavior. There are rules that belong to the control level because these are obvious physical level rules. Other rules belong to the agent level. Typically, if a rule is associated with a time critical action (actuation) of the physical device, it will be preferred to encapsulate this rule in the device driver and not in the agent. On the other hand, if a rule requires larger amount of information and time to determine and outcome, it will be preferred to create this rule in the agent level. Other rules of thumbs that also apply are:

1. Agents are specialized, can solve only problems from their domain;
2. Agents have preferably simple rules;
3. By combining simple rules there emerges a new more complex behavior; and
4. Due to emergent behavior, there is strong need for agent testing.

3.1 Designing Agents

Since the effort to program agents may become significant, a natural approach is to develop the notion of an "agent library", essentially a collection of class definitions that describe the behavior both of the agents and of any non-agent components.

Each library is targeted to a specific application domain, e.g., material handling systems or chilled water systems, and so can be used to create control systems for multiple similar facilities in the same domain. The control code is generated by the same process from the same templates. In this way, the effort to build the library is amortized over all the facilities built with the library, and each facility garner the benefits of having control system software that is structured, well defined, predictable, and tested. Moreover, any additions and/or changes to the library subsequently appear in all affected instances. In other words, instead of changing all affected instances individually, the user applies the change only in one place in the library, thereby not only speeding up the process but also avoiding any inconsistencies.

3.2 Behavior Description

The description of control behavior consists of a set of class descriptions for the various component classes. A component may or may not be an agent. If so, in addition to control behavior, it contains a description of its intelligent behavior.

The control behavior has been augmented to contain the description of intelligent behavior for the classes that have such behavior, i.e., for agents. Mainly, the additions include the capabilities and operations of the agents and the scripts that are used for planning.

3.3 Capabilities and Operations

A capability is a set of related behaviors or operations; each message received by an agent refers to exactly one operation. An operation is specified through its signature, i.e., its name plus its collections of inputs and outputs. An agent can support multiple capabilities. For example, for a chilled water system (CWS), some appropriate capabilities are: SupplyColdWater and PlanPath. The capability PlanPath has the following operations, written as (outputs) = operationName(inputs):

- (cost, path) = getPath(destination, currentCost, currentPath, nextDestinations, maximumCost)
- () = canAcceptWaterFrom(currentPath, nextDestinations, waterSource)

By registering with a certain capability, an agent supports all defined operations for that capability and thus can accept and process any message referring to one of these operations. All agent behaviors occur as a result of receiving a message. A message can come either from another agent or from the control portion of the same agent. The

possible behaviors that an agent can choose to execute upon receipt of a message are encoded in scripts. Upon receipt of a message, the agent chooses an appropriate script.

Each step within a script is either local or global. A local step performs action only on the agent's own data, e.g., reading one's own state, computing some value, or initiating some control action. A global action causes messages to be sent to other agent(s) in an attempt to delegate some portion of the agent's activity or to request information.

4 How to Support Distributed Agents

One of the key benefits of using multi-agent systems is their ability to work in a distributed environment. Agents use social skills to overcome this distribution. Therefore, it is necessary to ensure correct behavior of all communication layers and also to choose appropriate way of social knowledge distribution. From the point of view of multi-agent systems, it is possible to distinguish two main levels of communication:

- *Low level* communication must be reliant and safe. It must guarantee 100% message delivery, fragmentation and reassembly of messages, and it must avoid duplication of messages.
- *High level* communication resides directly on the low level communication. The system designer has to ensure that agents form and send messages correctly, that they are able to understand and correctly respond to incoming messages, etc. For instance, directory facilitator agent has to correctly propagate information about agent community members and their capabilities.

We use the FIPA Agent Communication Language (ACL) specification to build standard messages. This defines the encoding, semantics, and meaning of the messages. FIPA does not specify a mechanism for transportation of messages within an agent platform (i.e., low level communication), only declares that messages should be encoded in a textual form to overcome bridging into heterogeneous platforms.

In our industrial control implementation, the FIPA messages are encapsulated inside Common Industrial Protocol packets (CIP). This is fully compatible with industrial setups because it does not require special topology or dedicated physical links.

Social knowledge is concerned with the information about agents in the multi-agent system and consists of the name of agents, their location (address), their capabilities (services), the language they use, their actual state, their conversations, and behavioral patterns. Social knowledge can be distributed as follows:

- *Centralized* – stored, managed, and offered from one physical or functional point. For instance, blackboard architecture and federated architectures with one middle-agent.
- *Distributed* – statically or dynamically plugged directly into all agents in the system. For instance, architecture without any middle-agent.
- *Hybrid* – social knowledge is organized into groups, hierarchies, etc. For instance, teamwork-based technique and distributed matchmaking.

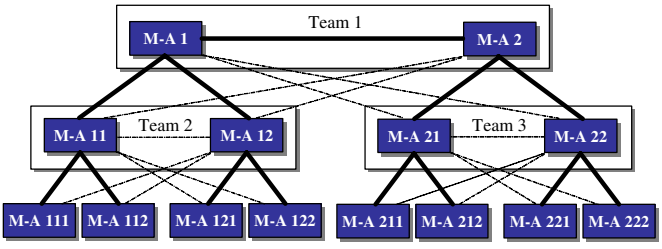


Fig. 3. Example of 3-level DHT architecture

We implemented a structure of middle-agents called dynamic hierarchical teams (DHT) [6]. This structure has user-defined level of fault-tolerance and is moreover fixed scalable, i.e., the structure can be extended by fixed known cost. Several approaches have been used already to deal with these issues. The teamwork-based technique uses a group of interconnected middle-agents, which forms a complete graph. A multi-agent system consists of middle-agents and end-agents. Middle-agents form a structure that can be described by the graph theory, as shown in Figure 3.

Graph vertices represent middle-agents and graph edges represent a possibility for direct communication between two middle-agents, i.e., communication channels. Each middle-agent that is not a leaf in the tree should be supported by another middle-agent. Whenever one of the middle-agents from the team fails, other middle-agents from the team can subrogate this agent. During the normal operation of the DHT structure all middle-agents use only primary communication channels (solid lines). If a failure of a primary channel occurs then a secondary channel (dashed lines) is used.

5 Validation System

Due to the emergent behavior, there is a strong need for agent testing to ensure that the whole system works properly and that emergent behavior is correct before formal delivery. Agent testing is done on simulators, not on the real hardware (to protect the real hardware and since the use of simulator speeds-up the whole development process). There is important work needed to create a reusable agent validation system. Herewith we explain some of the requirements for a successful implementation.

5.1 Synchronization

One goal is to synchronize controllers (SoftLogix or other control codes) with simulators (Matlab, Arena, or other simulators). The timing properties of the synchronization must be properly set to keep the dynamics of the simulated process. To provide this functionality, we split the whole process into two pieces: (a) Configurator, and (b) Synchronizer.

The configurator is used by the user to specify all modules and to enter all information needed for the synchronization. The configurator generates all necessary files, downloads projects into SoftLogix controllers, invokes instances of simulators and initializes them. The synchronizer provides a mechanism to properly synchronize

activities of all modules. The synchronizer runs controllers and simulations alternately and it ensures the data exchange between all modules.

Each module needs to be extended with a synchronization support. This additional code provides new functionality requested by the synchronizer and communicates with the synchronizer via generic interface. The generic interface is represented by the data structure located on the OPC server (OPC is well accepted industrial standard).

The controllers contain the agents and control programs. The simulation contains the physical plant simulation. There is an interface between the controllers and the simulations corresponding to the input and output signals (I/O) that will occur between the controllers and the physical devices. Since this I/O interface is official, it will not change when commissioning the software from the validation phase into the deployment phase. Thus the efforts put in validating the agent and control is compensated by this smooth transition from the lab into the plant. The caveat is that the validation system depends on the availability of a good simulation of the plant.

5.2 How the User Operates the System

The user configures the whole system using the configuration tool. The configurator allows the user to add or remove a module from the simulation, set the time parameters of the synchronizer, and link data tags in modules. Each module (SoftLogix, Matlab, or other simulator) has attached one data structure for synchronization. The data structure is located on the OPC server. The structure serves as a generic interface to each module.

The engineering requirements for a validation system would include at least: (1) Access to control configuration, (2) Automatic configuration and downloading to interconnect simulations with controllers, (3) Mapping of simulation and control tags, (4) Clock synchronization, and (4) OPC or other standard data exchange.

6 Final Remarks

A strategy to create viable agent based systems for industrial deployment requires multiple disciplines. It is desirable to create the infrastructure and platforms for modeling and validation in a generic manner. It is recommended that the agent design tools hide the burden of programming agents as much as possible and that the tools present the programming of the agents in a language understandable by common users. This is difficult to achieve independently. However, we believe that by using standard and international specifications such as FIPA, CIP, IEC 61131, IEC 61499, OPC, etc. this endeavor will be possible. Nevertheless, there is no unique philosophy to carry out agent design and validation.

References

1. Christensen, J.H.: Holonic Manufacturing Systems: Initial Architecture and Standards Directions. First European Conference on Holonic Manufacturing Systems, Hanover, Germany (1994)
2. FIPA: The Foundation for Intelligent Physical Agents, Geneva, Switzerland, 1997.

3. Mařík, V., Pěchouček, M., Štěpánková, O.: Social Knowledge in Multi-Agent Systems. In Multi-Agent Systems and Applications, LNAI 2086, Springer, Berlin (2001) 211-245
4. Maturana F.P., Staron R., Hall K.: "Methodologies and Tools for Agents in Distributed Control". In IEEE Intelligent Systems Magazine, pp. 42-49, January/February 2005.
5. Leitão P. and Restivo F., A Holonic Control Approach for Distributed Manufacturing, In: Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle, V. Marik, L.M. Camarinha-Matos and Hamideh Afsarmanesh (eds), pp 263-270, Kluwer Academic Publishers, 2002.
6. Tichý P.: Middle-agents Organized in Fault Tolerant and Fixed Scalable Structure. In Computing and Informatics, ISSN 1335-9150, Vol. 22, pp. 597-622, 2003.
7. Shen W., Norrie D., and Barthès J.P.: "Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing". Taylor & Francis, London, 2001.
8. Vasko D., Maturana F., Bowles A., and Vandenberg (2000): Autonomous Cooperative Systems Factory Control. PRIMA 2000, Australia, 2000.
9. Discenzo F., Maturana F., and Chang D., Managed Complexity in an Agent-Based Vent Fan Control system Based on Dynamic Reconfiguration, International Conference on Complex Systems, June 9-14, 2002, Nashua, NH.
10. Maturana F., Balasubramanian S., and Vasko D.: An Autonomous Cooperative System for Material Handling System Applications, ECAI 2000. Proceedings of the 14th European Conference on Artificial Intelligence, IOS Press, Amsterdam, 2000.

Experimental Validation of ADACOR Holonic Control System

Paulo Leitão[1] and Francisco Restivo[2]

[1] Polytechnic Institute of Bragança, Quinta Santa Apolónia, Apartado 1134, P-5301-857 Bragança, Portugal
pleitao@ipb.pt
[2] Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, P-4200-465 Porto, Portugal
fjr@fe.up.pt

Abstract. In the last years, several manufacturing control architectures using emergent paradigms and technologies, such as multi-agent and holonic manufacturing systems, have been proposed to address the challenge of developing control systems capable of handling certain types of disturbances at the factory level. One of these holonic architectures is ADACOR, which integrates a set of paradigms and technologies for distributed manufacturing systems complemented by formal modelling techniques, to achieve a flexible and adaptive holonic/collaborative control architecture. The results obtained in the first experiments using the ADACOR architecture are presented in this paper, and also compared to the results produced by other control architectures. For this purpose a set of quantitative and qualitative parameters were measured, to evaluate static and dynamic performance of the control architectures.

1 Introduction

In general, manufacturing systems are heterogeneous environments, comprising a variety of hardware and software applications. They are also asynchronous, stochastic and dynamic environments, with certain resources becoming unavailable and additional resources being introduced at random times, new jobs arriving continuously to the system, new products being frequently defined, and new regulations, such as quality and safety specifications, being regularly announced.

The economical and technological trends, associated to the customer satisfaction and products shorter life cycle impose new requirements to manufacturing systems that lead to new organizational structures (distributed, dynamic and open), flexibility and agility to support volatile and dynamic markets.

To face these requirements the new generation of manufacturing control systems should exhibit important features, such as agility (reacting rapidly to the occurrence of disturbances), re-configurability (changing dynamically its configuration, without stopping or re-starting the process), scalability (accepting the addition of new components, without the need to re-design, re-program or re-initialize the existing ones), re-usability (allowing to re-use past or previous

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 121–132, 2005.

solutions to simplify development) and intelligence (anticipating future demands and learning from the past experience).

In the last years, several manufacturing control architectures using emergent paradigms and technologies, such as multi-agent and holonic manufacturing systems, have been proposed to address this challenge (see [1,2,3,4]). One of these holonic architectures is ADACOR (ADAptive holonic COntrol aRchitecture for Distributed Manufacturing Systems) [5], which integrates a set of paradigms and technologies for distributed manufacturing systems (HMS, MAS, ...) complemented by formal modelling techniques (Petri nets, AUML, ...), to achieve a flexible and adaptive holonic/collaborative control architecture.

ADACOR holonic control system is built upon a set of autonomous, cooperative and self-organized holons, each one representing a manufacturing component. ADACOR defines four holon classes [5]: product, task, operational and supervisor. The product holons represent the products available in the factory catalogue, the task holons represent the production orders launched to the shop floor and the operational holons represent the physical resources available in the shop floor. The supervisor holons provide co-ordination and optimization services to the holons under their supervision.

The adaptive ADACOR production control approach is neither completely decentralized nor hierarchical, but balances between a more centralized and a more flat approach, passing through other intermediate forms of control [5]. The presence of supervisor holons in a decentralised system, and the presence of self-organization capability associated to each ADACOR holon allows the evolution or the re-configurability of the control system, combining the global production optimization with the agile reaction to unpredictable disturbances.

ADACOR production control evolves in time between two alternative states, stationary and transient states [5]. In stationary state the holons are organized in a hierarchical-like structure, with supervisor holons coordinating several operational and/or supervisor holons. The role of each supervisor holon is to introduce global optimization in the production process. The transient state, triggered with the occurrence of disturbances, is characterized by the re-organization of the holons in a heterarchical-like control architecture, allowing the agile reaction to disturbances. This re-organization is performed through the self-organization of each holon, mapped with the increase of its autonomy and the propagation of the disturbance to the neighbor holons using ant-based techniques. After the disturbance recovery, the operational holons reduce their autonomy, evolving the system to a new control structure (often returning to the original one).

The validation of ADACOR concepts requires their implementation in a real environment, to analyze their correctness and applicability. In spite of the promising perspectives of the holonic manufacturing paradigm and the research developed by the holonic community, only few industrial implementations were reported in the literature, such as those described in [6,7]. This paper describes the experimental validation of ADACOR concepts in a flexible manufacturing system. The experimental results extracted from the implementation and testing allows to evaluate the ADACOR control system performance, both in terms of

quantitative indicators directly related to production parameters and of qualitative indicators related to the dynamical behaviour of the system.

2 Experimental Case Study

A pilot installation has been used to validate the ADACOR holonic control system, aiming to address two main objectives: i) validate the concepts and the implementation, i.e. to verify if the system works as it was specified, and ii) evaluate the performance to conclude about the merits of the proposed concepts. The case study used in this work will be described in the next sections.

2.1 Pilot Installation

The pilot installation is a semi-virtual laboratorial platform based on the flexible manufacturing system of the CIM Centre of Porto, which is 'extended' for our purpose with two virtual manufacturing cells, to provide the necessary hardware/software redundancy and flexibility in accommodating alternative solutions at the production planning level, as illustrated in Figure 1. The flexible manufacturing platform is organized as a set of four physical cells: manufacturing cell, assembly cell, storage and transportation cell and maintenance and setup cell. Cells B and C in the figure do not exist in the real platform.

The assembly cell is responsible for the assembly of the components to achieve the final products. This cell has a SCARA robot Adept Three from Adept Technology. Coupled to the robot, there is a CCD camera from PULNIX, associated to an artificial vision system Cognex 4200EX from Cognex Corporation.

The storage and transportation cell is responsible for the transportation of materials within the shop floor and for the temporary storage of materials.

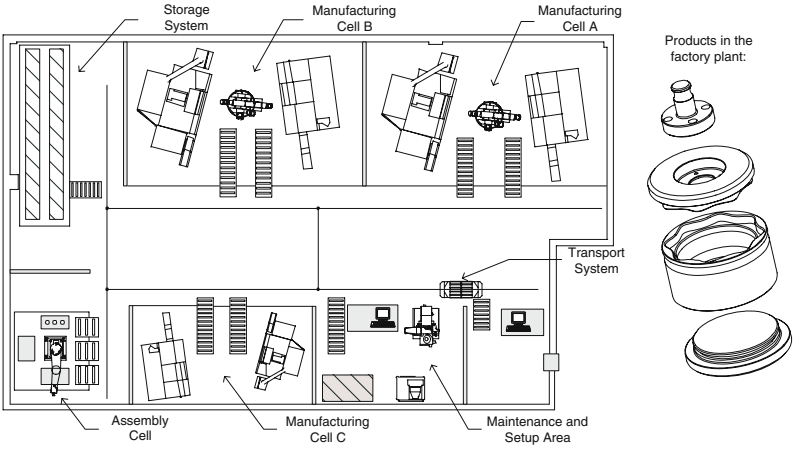


Fig. 1. Plant layout of the case study production system

This cell has an AGV (Automatic Guided Vehicle) and an Automated Storage/Retrieval System. The presence of the AGV allows variable routing of the products flow.

The maintenance and setup cell is responsible for maintenance, setup and recovery operations, assembly of tools, calibration of tools and grippers, and palletizing and demagnetizing of the materials that circulate in the shop floor. This cell includes a tool calibration system AR2000GA from Elbo Controlli, a palletising table and several equipments to support the maintenance operations.

The manufacturing cell has two CNC machines and one anthropomorphic robot for the load/unload of these machines. One of these machines is a turning center Lealde TCN10, with a SIEMENS Sinumerik 880T controller. The other machine is a milling center Kondia B500, with a FANUC 16MA numerical control. The robot is a KUKA IR163/30.1 with a SIEMENS RC3051 controller.

The production system contains other types of resources, namely buffers and containers. Each machine has its input/output buffer, to de-couple it from the transport system. The containers bring the material to be processed in the machine or the cell and take away the pieces produced.

In this pilot plant, four different (sub-)products are produced, as illustrated also in Figure 1, named Base, Body, Cover and Handle. When assembled, they can create two different final products: Box and Ashtray. The Ashtray product comprises the assembly of the Base and the Body sub-products, and the Box product comprises the assembly of all designed sub-products.

2.2 Manufacturing Scenarios

The experiment considers three different plant scenarios: i) the first plant scenario considers that no unexpected disturbance will occur, ii) the second plant scenario considers the occurrence of failures in one turning machine (cell B), with a probability of 25%, and that, in case of failure, the part is destroyed and the machine is down during 60 seconds for the recovery procedures, and iii) the third plant scenario considers the occurrence of failures in the turning machines of cells B and C, with the same disturbance model of the previous scenario.

In this experimental test it is considered that no setups are executed, since machines are equipped with the required tools to execute a range of operations. The transport operations are performed by a single AGV and orders are queued by order of arrival. The execution of each transport operation takes 5 seconds.

Each individual book of orders comprises the production of 6 production orders: 2 bodies, 2 bases, 1 handle and 1 cover. The experimental test reported in this paper considers a plant load of three books of orders, i.e. 18 production orders (involving 51 operations). All the production orders belonging to the same book of orders arrive to the production system at the same time, but different books of orders arrive sequentially to the production system.

Each experiment was executed 6 times, and average values and standard deviations were computed. As the control systems run on a multitask platform with parallel threads and inter-process communications, originating a certain degree of randomness in the allocation of operations, and thus producing some

stochastic variations in the processing times, more accurate results are obtained this way.

2.3 Performance Indicators

Performance indicators can be classified as qualitative or quantitative. The quantitative indicators are based on different production performance measures, such as the lead time and throughput. The qualitative indicators are of a more subjective nature and reflect properties of the manufacturing control solution, such as the agility and flexibility, which cannot be directly obtained from the production data. In this experimental test, the ADACOR holonic control system was evaluated by analyzing the following performance indicators: manufacturing lead time, throughput, repeatability, resource utilization and agility.

The manufacturing lead time is the total time required to process a given product through the factory plant, and comprises the setup time, the no-operation time, the idle time and the processing time. The shorter the lead time is, more products can be produced by the production plant in the same period of time. The lead time reflects factory plant optimization level and productivity.

The throughput is an indicator of the productivity of a manufacturing system, and is defined here as the number of items produced per time unit. In the context of this work, the throughput was measured as the ratio between the number of parts produced in the experience and the batch time.

The resource utilization is defined as the percentage of processing time during a time interval. The average resource utilization is equal to the mean value of the percentage of utilization of all resources in the system. The analysis of the standard deviation of the utilization of all the resources allows to verify if the manufacturing load is evenly distributed by all the resources or concentrated in a few ones. A high value for this parameter may indicate the existence of overloaded resources, and the need to re-allocate some load to other resources.

The repeatability of the manufacturing control system is given by the mean value of the standard deviation of the percentage of utilization of all resources of the system over the several experiences. The smaller the repeatability is, more repeatable is the manufacturing control system production plan.

The agility of a control system can be defined as the capability to react in a short period of time to the occurrence of unexpected disturbances, more exactly, the time needed by the system to recover properly from the occurrence of a disturbance.

In this experimental test, the agility parameter is evaluated by running n experimental tests and analyzing the loss of productivity in presence of disturbance scenarios. For that purpose, it is necessary to know in first place the time required to produce a specific amount of items with no disturbances. Then, it is measured the time required to execute the same products, under a disturbance scenario. Having these two values, it is possible to calculate the throughput in each case, and the percentage of reduction of throughput, which is the loss of

productivity. The loss of productivity reflects indirectly how agile the system is. The smaller the loss of productivity value is the higher the agility of the system will be.

3 Prototype Implementation and Operation

The ADACOR control system prototype was implemented using multi-agent systems technology [8]. From the set of available commercial and academic agent development platforms [9], it was chosen the JADE (Java Agent Development Framework) platform (see http://jade.cselt.it/), which complies with the FIPA (Foundation of Intelligent Physical Agents) specifications.

3.1 Implementation

An ADACOR holon is a Java class that extends the Agent class provided by the JADE framework, inheriting basic functionalities, such as registration services, remote management and sending/receiving ACL messages, and adding features that represent the specific behaviour of each ADACOR holon class. The behaviour of each ADACOR holon uses multi-threaded programming, over the concept of JADE's behaviour, to allow the execution of several actions in parallel. The set of behaviours (each one corresponding to a kind of thread) launched at the start-up and those that can be invoked afterwards are provided in the form of Java classes.

The communication between distributed holons is done over the Ethernet network, the messages being encoded using the FIPA-ACL communication language. The content of the messages is formatted according to the FIPA-SL0 language and the meaning of the message content is standardized according to the ontology defined by the ADACOR architecture.

The decision component of each ADACOR holon uses declarative and procedural approaches to represent knowledge and to regulate the holons behaviour [8]. The central element in the decision component is the rule-based system developed using JESS (Java Expert System Shell), which applies declarative knowledge expressed in a set of rules. ADACOR holons uses also procedural knowledge embodied in procedures that are triggered as actions by some rules, each one being responsible for the execution of a particular set of actions. The scheduling algorithm is an example of this type of knowledge representation.

The implementation of operational holons that represent physical automation resources requires the development of wrapper interfaces, supporting the integration of those resources within the holon. ADACOR introduces the virtual resource concept making transparent the intra-holon interaction [8]. The development of a virtual resource for each manufacturing device encompasses the implementation of the services at the server side (the real automation resource), which will be invoked on the client side (the logical part of the operational holon). The client ignores the details of this implementation and each virtual resource can be re-used by other similar resources or holonic control applications.

3.2 Operation

A Factory Plant Supervisor tool was developed to monitor, in an integrated way, the production activities in the factory plant. This tool, represented in Figure 2, allows visualizing the state of the manufacturing resources present in the factory plant. The graphical animation associated to the AGV helps to understand the material flow in the factory plant.

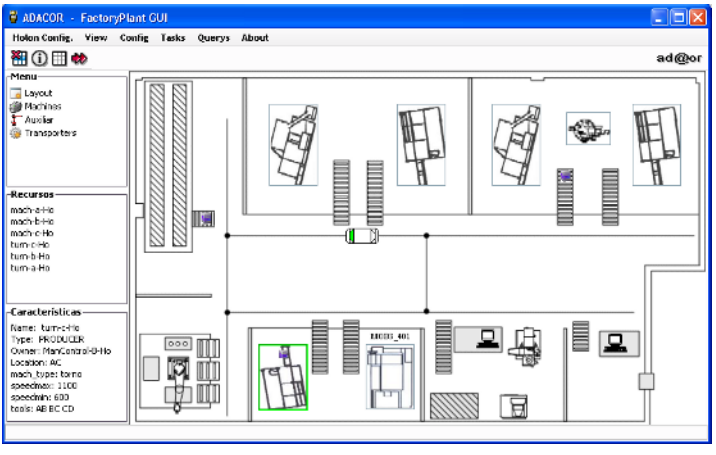


Fig. 2. Screenshot of ADACOR Factory Plant Supervisor

During the experimental test the several holons presented in the system were distributed by different PCs, running in different platforms such as Windows XP, Windows 2000 and Linux. This allowed demonstrating that ADACOR control system supports the heterogeneity presented in industrial automation scenarios.

The experience gained during the prototype implementation, debugging and testing allows extracting some conclusions about the operation of ADACOR holonic control system. In a first instance, it was verified that it works as specified, either in normal operation or in presence of disturbances. This was one of the major objectives of the experimental validation.

Additionally, the re-configurability of the ADACOR holonic control system was proven, since the system reacted correctly to the introduction, removal and modification of manufacturing components. Specially, it was shown that when a supervisor or operational holon breaks down or leaves the system, the other holons continue their way finding alternative solutions to execute the production plan. This is mainly supported by the plug-and-produce characteristic associated to the ADACOR holons, i.e. each ADACOR holon works autonomously, not requiring the need for additional re-design, re-program and re-start of other components.

4 Analysis of Quantitative Performance Indicators

The first set of experimental tests evaluates the behaviour of the ADACOR control approach, by comparing its performance with the hierarchical and heterarchical control approaches, focusing in the analysis of the quantitative indicators.

An important remark is related to clarify that all the three control approaches uses the same prototype platform: i) in the hierarchical control approach, the holons are organized in a hierarchical control structure, using supervisor holons, ii) in the heterarchical control approach, the holons run on a completely decentralized control structure, without the presence of supervisor holons, and iii) in the ADACOR holonic control approach, the holons are organized in a hierarchical control structure, using supervisor holons and enabling the self-organization of each operational holon to support the agile re-organization of the control structure in case of emergency.

4.1 Stable Scenario

In a scenario without the presence of unexpected disturbances, the system operates in a predictable way. The results of this experimental test are summarized in the Figure 3. In a stable scenario the holons of the ADACOR control approach are organized in a hierarchical structure, presenting the same behaviour as in the hierarchical-like control, therefore showing the same experimental values.

In stable scenarios the hierarchical-like and ADACOR control approaches present smaller values of manufacturing lead time (336,2) and higher values of the throughput (49,4) than the heterarchical-like control approach (respectively 387,2 and 46,0). The better performance presented by those approaches results from the better production planning achieved by the centralized entities, i.e. a supervisor holon that elaborates optimized production plans.

Analyzing the repeatability of the production planning, which is a measure of its predictability, it is clear that the repeatability in hierarchical-like and ADACOR control approaches is better than in the heterarchical-like control approach. In fact, in the heterarchical-like control approach the global schedule is achieved by the interaction of operational holons that have a partial view of the entire system, making difficult to achieve the global optimization.

The type of production has also strong impact in the degree of production optimization achieved. It was verified that for operations with short processing

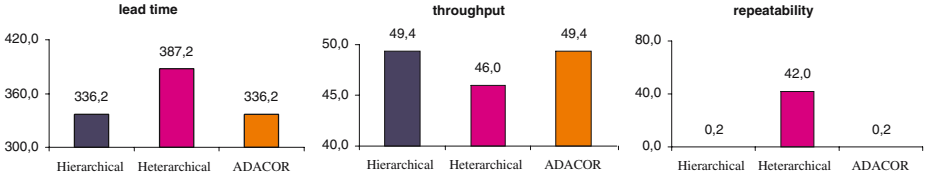


Fig. 3. Performance of evaluated control approaches for scenarios with no disturbance

times, the better performance resulting from the global optimization present in hierarchical-like and ADACOR control approaches is less visible, while in cases of operations with longer processing times, the non-optimised schedules present in heterarchical-like control approaches lead to even worse performance. This is due to the fact that longer processing times are more sensitive to weak global optimization.

4.2 Disturbance Scenarios

The second experimental test considers the occurrence of unexpected disturbances in the turning machine of cell B, according to the probabilistic disturbance model described in section 2.2. The results obtained in this experimental test are summarized in the Figure 4.

The first conclusion extracted from these experimental results is the degradation of all performance indicators in the presence of disturbances.

Analysing the lead time and throughput parameters, it is possible to verify that the hierarchical-like control approach still presents better performance than the heterarchical-like control approach. However, it is possible to verify that the difference of performance between hierarchical-like and heterarchical-like control approaches has been significantly reduced, specially in terms of the throughput parameter (reduction of 8,7% for the lead time and 55,9% for the throughput).

The occurrence of disturbances increases the entropy and unpredictability of the control system. It was verified that in disturbance scenarios the differences between the predictability exhibited by the several evaluated control approaches are smaller (i.e. between 52,9 to 66,9). Thus, the proposed ADACOR holonic control approach presents promising performance results, since it shows better response to the disturbance scenario, illustrated by smaller value of manufacturing lead time (337,7) and higher value of throughput (46,6), than the hierarchical-like and heterarchical-like control approaches.

The second disturbance model was used to compare the response of the three control approaches to the different levels of entropy caused by the occurrence of disturbances. In this scenario, it is assumed that failures can occur in turning machines of cells B and C. Figure 5 illustrates the results obtained during the execution of this experimental test.

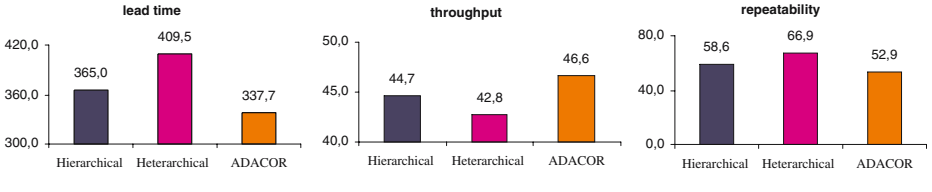


Fig. 4. Performance of evaluated control approaches for disturbance scenarios

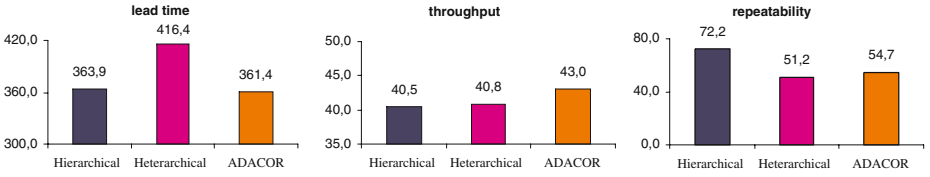


Fig. 5. Performance of evaluated control approaches for the 2nd disturbance model

The experimental results confirm the observations done during the previous experimental test, being clear that the performance of each control approach suffers with the increase of entropy associated to the disturbance model.

4.3 Analysis of Resource Utilization

For the purpose of the resource utilization analysis, only the three turning machines were considered, since they are the ones that provide alternative paths to execute the operations to manufacture the package of available products. The average and the standard deviation of the percentage of resource utilization, for the stable and first disturbance scenarios are summarized in the Figure 6.

Analyzing the experimental results, it is possible to verify that the hierarchical-like and ADACOR control approaches present higher percentage of resource utilization than the heterarchical-like control approach, either in the stable and disturbance scenarios, which demonstrates their better production plan optimization. It can also be observed that the percentage of resource utilization is higher in disturbance scenarios than in the stable scenario, due to the execution of additional work orders launched to the shop floor, after the occurrence of machine failures that destroyed the part.

The analysis of the standard deviation of the percentage of resource utilization, which gives an idea about how the planning and control system distributes the load by the available resources, allows verifying that the heterarchical-like control approach presents the worse behaviour in the load distribution variability, as expected. On the other hand, the ADACOR control approach presents values for the standard deviation that are even smaller than those for the hierarchical-like control approach.

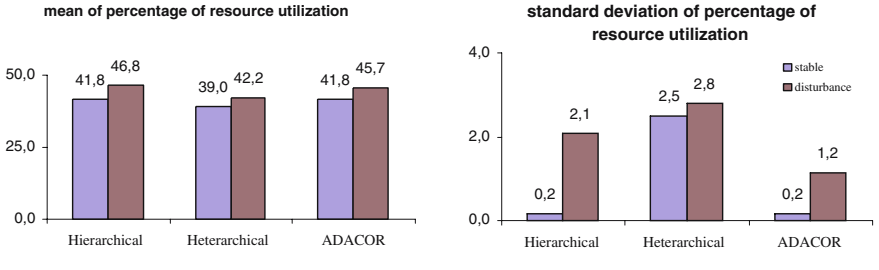


Fig. 6. Experimental results of the resource utilization

5 Analysis of Qualitative Performance Indicators

In this section, the ADACOR control system is evaluated by analyzing a single qualitative performance parameter, the agility. The comparison of the loss of productivity values for the two disturbance models is illustrated in Figure 7.

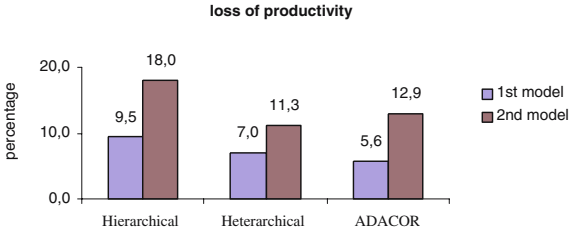


Fig. 7. Loss of productivity of the evaluated control approaches

It is possible to verify that the ADACOR control approach presents similar values to those exhibited by heterarchical-like control approach. As expected, the hierarchical-like control approach presents the higher loss of productivity. As the agility is inversely proportional to the loss of productivity, the experimental results show that the ADACOR control approach exhibits the same levels of agility of those of the heterarchical-like control approach. In scenarios where disturbances are more frequent, the levels of agility presented by the several control approaches are reduced.

Analyzing simultaneously the agility and the manufacturing lead time, the results obtained in these experimental tests reveals that the ADACOR holonic control system presents promising performance results, since it shows better response to the disturbance scenario, illustrated by smaller value of manufacturing lead time, and confirm that the ADACOR holonic control system combines the hierarchical and heterarchical best features, presenting similar values of agility to the heterarchical approach, but better production optimization.

6 Conclusions

ADACOR architecture addresses the agile reaction to the occurrence of unexpected disturbances at shop floor level, by introducing an adaptive production control approach, that evolves dynamically through different control structures supported by the self-organization capability associated to each ADACOR holon.

The results obtained in the first experiments using the ADACOR architecture were presented, and compared to the results produced by other control architectures. Three scenarios were devised, with different levels of occurrence of failures, using the same book of orders. Mean value and standard deviation of a set of quantitative and qualitative parameters were measured, to evaluate static and dynamic performance of the control architectures.

It was possible to conclude first of all that ADACOR architecture was able to handle all the possible combinations of situations, allowing concluding that the ADACOR concepts are sound and ready to be used in real situations.

Performance measures also showed that ADACOR control architecture exhibits the better performance in what concerns to dynamical behaviour of the production facility after the occurrence of failures.

An important issue to consider in future work within ADACOR is to focus in the performance measurement of manufacturing control systems. Some ongoing work in this issue is being done by Intelligent Manufacturing Systems Network of Excellence (IMS-NoE), Special Interest Group on Benchmarking and Performance Measures of on-line Production Scheduling Systems (see http://www.ims-noe.org), where the authors are associate members.

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, pp. 255-274.
2. Deen, S.M. (ed.): Agent-Based Manufacturing: Advances in the Holonic Approach, Springer Verlag Berlin Heidelberg, 2003.
3. Maturana, F. and Norrie, D.: Multi-Agent Mediator Architecture for Distributed Manufacturing. In: Journal of Intelligent Manufacturing, vol. 7, 1996, pp.257-270.
4. Chirn, J.-L. and McFarlane, D.: A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture. In: Proc. of the Intern. Workshop on Industrial Applications of Holonic and Multi-Agent Systems, 2000, pp. 219-223.
5. Leitão, P., Colombo, A.W. and Restivo, F.: ADACOR, A Collaborative Production Automation and Control Architecture. In: IEEE Intelligent Systems, 20(1), 2005, pp. 58-61.
6. Colombo, A. W., Schoop, R. and Neubert, R.: Collaborative (Agent-Based) Factory Automation. In: The Industrial Information Technology Handbook, R. Zurawski (ed), CRC Press, 2004.
7. Maturana, F., Staron R., Tichy, P. and Slechta, P.: Using Dynamically Created Decision-Making Organisation (Holarchies) to Plan, Commit and Execute Control Tasks in a Chilled Water System. In: Proceedings of the 3rd Int'l Workshop on Industrial Applications of Holonic and Multi-Agent Systems, 2002, pp. 613-619.
8. Leitão, P., Casais, F. and Restivo, F.: Holonic Manufacturing Control: A Practical Implementation. In: Emerging Solutions for Future Manufacturing Systems, L. Camarinha-Matos (ed.), Springer Science+Business Media, 2004, pp. 33-44.
9. Vrba, P.: JAVA-Based Agent Platforms Evaluation. In: Holonic and Multi-Agent Systems for Manufacturing, LNAI 2744, Springer-Verlag, 2003, pp. 47-58.

A Proxy Design Pattern to Support Real-Time Distributed Control System Benchmarking

K. Soundararajan and R.W. Brennan

Department of Mechanical and Manufacturing Engineering,
University of Calgary, 2500 University Drive NW,
Calgary, Alberta T2N 1N4, Canada
rbrennan@ucalgary.ca

Abstract. In this paper we propose a hybrid physical/simulation environment for benchmarking real-time distributed control systems. This environment uses Arena® Real Time for simulation and the Java-based TINI platform for real-time control. The paper focuses on the development of a software design pattern for client-server communication.

1 Introduction

Given the difficulty of practical manufacturing scheduling and control problems, recent research has moved away from traditional, analytical approaches that have been the domain of operations research for many years and towards new approaches that rely on artificial intelligence, holonic and multi-agent systems (Shen et al., 2001; McFarlane and Bussman, 2000). In order to make this research relevant however, it is important that realistic and industrially relevant test cases (benchmarks) are available to address specifically the evaluation and stress of the performance of scheduling and control systems based on these new technological paradigms. As well, it is important that these test cases span the realm of research in this area from planning and scheduling systems to real-time control.

In this paper, we describe a hybrid physical/simulated environment that is currently being developed for manufacturing systems control experimentation that incorporates both simulated and physical manufacturing devices. In order to accomplish this, an important aspect of the project involves developing an interface between the simulation software and the physical devices. In our work, we are currently investigation the use a Tiny Internet Interface (TINI) board (Loomis, 2001) that runs Java programs (allowing us to develop local software), and has access various I/O (e.g., discrete/analog I/O, Ethernet). The real-time control applications on the TINI board are developed using IEC 61499 function block applications (IEC, 2000), while the discrete-event simulation model used for this research is Arena® Real Time (Kelton et al., 1998).

This paper focuses on the software interface between the physical device and the simulation, and in particular the ability to abstract the true server from the client by means of a "stand-in" or surrogate class. In this case, our server is the discrete-event simulation model and the client is the physical device: the goal is to abstract the functions of the different components of the distributed system. In particular, we would like to separate those functions specific to the embedded controller with those

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 133 – 143, 2005.

required for inter-process communication with the remote server. On the server side, the communication functions with the client embedded controller need to be separated from the discrete event simulation model workings.

We begin the paper with an overview of this hybrid simulation/physical environment in Section 2 and details of its implementation in Arena® and IEC 61499 in Section 3. Next, we focus on the communication interface and setup the proxy pattern problem for this case. The paper then focuses on a description of the proxy pattern that was developed for this work and the consequences of its uses for a hybrid physical/simulation environment as well as its potential uses for other similar distributed control problems.

2 A Hybrid Simulation/Physical Environment

In the manufacturing domain, discrete-event simulation is a very powerful tool that can be used to evaluate alternative control policies. For example, discrete-event simulation has been used to evaluate agent-based scheduling approaches by interfacing agent-based or object-oriented software with a discrete-event simulation model of a plant to be controlled as is illustrated in Figure 1 (Brennan and O, 2004).

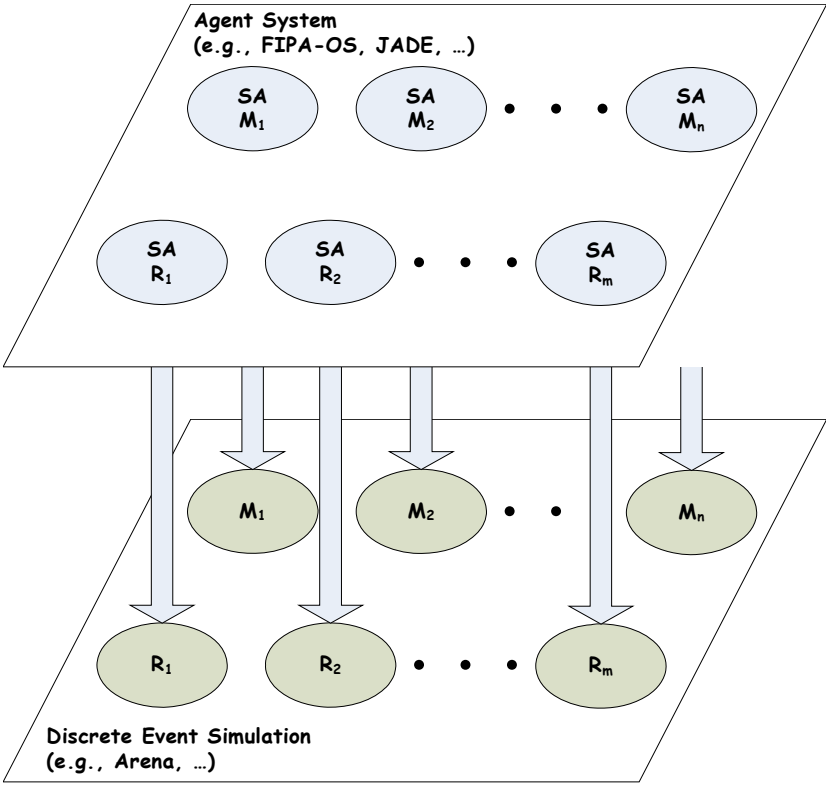


Fig. 1. Using discrete-event simulation to evaluate alternative agent-based control policies

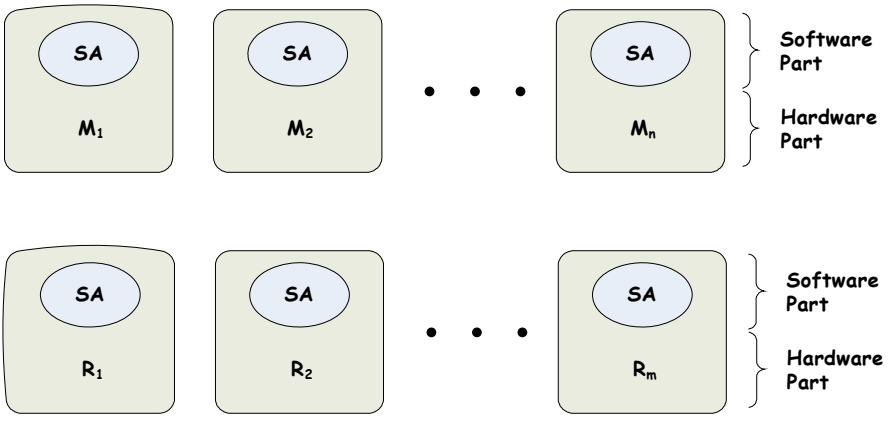


Fig. 2. Physical agents

In this example, each entity in the discrete-event simulation model (such as a machine (e.g., M_1) or robot (e.g., R_1)) is represented by a corresponding software agent (SA) in the control module. For example, the software agents can be thought of as the "reasoning" part of the entity that is responsible for scheduling etc.

The reason for this direct correspondence between SA and entity is that (given recent advances in hardware and software) is it possible to have intelligent agent software running directly on a machine (e.g., computer numerical control (CNC), robot, etc.). This software can be thought of as the "brains" of the machine that can potentially allow it to act autonomously and/or cooperatively.

The next step is to have these SA's run directly on the machines. This will allow us to test the real-time capabilities of the system (i.e., its ability to meet deadlines), and also allow us to incorporate extra functionality concerned with "execution". For example, SA's (running directly on a machine) may be used to perform fault diagnosis and preliminary recovery services. SA's may also be capable of reconfiguration (e.g., allowing new hardware to be added/removed/modified dynamically). This arrangement is shown in Figure 2.

From an experimental and research point of view, there are some problems with this second approach even though it represents the ultimate goal (i.e., industrial implementation) of the agent-based system. The main problem is that, given financial and space resources, most researchers using this approach are limited to experimenting with relatively small systems. As well, even if a large system is possible, it is debatable whether it would be a good time and cost investment. For example, we may only need a relatively small number of physical devices to test and validate the execution capabilities of our agent system. However, we would like to have a reasonable number of emulated machines to test the scheduling capabilities.

The former requirement (execution) is typically hard real-time (i.e., deadlines must always be met very quickly), while the latter requirement (planning, scheduling, and dispatching) is typically soft real-time (i.e., deadlines must be met on average and much more slowly). As a result, we need physical hardware to test the execution environment and could use a simulated environment to test the "higher reasoning"

part of the system. Of course, physical hardware could also be used to test this latter part of the system.

A second problem (from a research perspective) with a pure physical system is that we lose many of the experimental benefits of discrete-event simulation software (e.g., statistical analysis, graphics, the ability to easily change the system configuration).

As a result, we suggest that a hybrid physical/simulated environment is developed for manufacturing systems control experimentation. In order to accomplish this, one aspect of the project involves developing an interface between the simulation software and the physical devices. One approach is to use a Tiny Internet Interface (TINI) board. This board runs Java programs (allowing us to develop local SA's), has access to discrete/analogue I/O, and has Ethernet capabilities. The general idea is illustrated in Figure 3.

In this example, M_1, M_2, and P_i have software agents associated with them (SA_1, SA_2, and SA_i respectively). M_3 and M_4 are also part of the system, but they are real physical devices. For example, the TINI boards could act as controllers for robots or

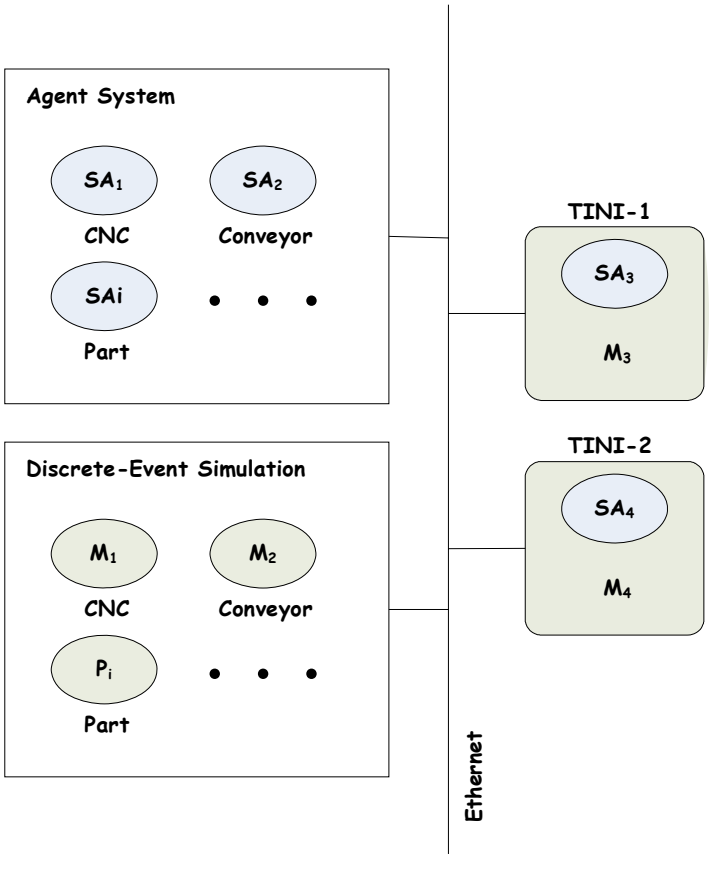


Fig. 3. A hybrid simulation/physical environment for experimentation

conveyors. Their agents (SA_3 and SA_4 respectively) communicate with the other parts of the system via a UNIX socket.

In this paper we focus on the interface between the discrete-event simulation and the physical controllers ("TINI-1" and "TINI-2"). However, the same interface can be used between the discrete-event simulation and the agent system and/or the agent system and the physical controllers. For example, when the discrete-event simulation is replaced by the physical manufacturing equipment, the physical equipment would be implemented by programmable logic controllers (PLC) or other embedded controllers and would communicate directly with the agent system (which may also be implemented on PLC's).

3 Implementing the Hybrid Environment with Arena and IEC 61499

In this section we begin with a brief overview of the simulation software used for the hybrid environment. This software is well suited to the needs of this research because of its ability to operate in a real-time mode, allowing the simulation to synchronise with the physical parts of the system. Next, we describe the mode of communication used between this simulation software and our physical devices.

3.1 Arena Real-Time

Arena Real Time (RT) is the real-time version of the Arena® suite of discrete-event simulation software environments (Kelton et al., 1998). Although the Arena RT version of the software is required for full real-time support, all of the real-time features are provided in the Standard Edition of Arena®. The only limitation is that Arena RT only allows simulation models to be run for a limited time (10 minutes).

Arena RT runs the simulation model in execution mode. When in this mode, Arena can coordinate simulation logic with an external process of a real system. The external processes and Arena communicate via a messaging system, whereby entities in the Arena model send messages to the external applications to indicate simulated tasks, and the external applications send "message responses" back to Arena to indicate the tasks have been completed. Unsolicited messages can also be sent to Arena to indicate special events (e.g., the arrival of raw material or customer orders). The Arena simulation clock speed is set to the real-time clock speed of the computers operating system.

The Arena RT extension is used in this research to coordinate a benchmark simulation model with the process of the TINI board described previously. This setup allows us to assess the hybrid model and the control software running on the TINI Board.

Two alternative programming approaches for implementing inter-process communication can be used with Arena RT: Visual Basic for Applications (VBA) events, or routines provided in SIMAN's C++ user-code library. In each case, the following basic functionality can be accessed from within the simulation model:

- Initialization – At the beginning of the first simulation replication, inter-process communication (e.g., opening a communication socket) is initialized.
- Message Transmission – This part of the system allows simulation entities to send messages to the external process via Arean's inter-process communication (IPC).
- Message Receipt – This part of the system allows messages to be passed from the IPC queue to Arena simulation entities.
- Termination – At the end of the last simulation replication, inter-process communication is terminated (i.e., the communication socket is closed).

In order to implement inter-process communication in Arena RT, the appropriate VBA code or C++ code (in the form of a DLL) must be linked to the Arena model. Typically, the Arena model acts as a server and the external processes act as clients.

3.2 Client-Server Communication

In this section we provides an overview of the client-side communication program that was developed in Java to allow communication between the Arena simulation model and the physical devices illustrated in Figure 3.

The TCP/IP protocol suite is used in this case to implement the basic network communication. As well, as noted previously, the mode of communication between the distributed applications used is the client/server mode. In this mode of communication, a client program initiates communication while the server program waits passively for and then responds to clients that contact it. As a result, the client program needs to know the server's address and port initially, but not visa versa.

For this application, Arena uses a C++ program to establish Arena as a TCP/IP server. The server's job is to set up a communication endpoint and passively wait for connections from clients. In this case, the TCP server first constructs a socket instance at a specified port (this socket listens for incoming connections to the specified port). Next, the server repeatedly calls "accept" methods to get the next incoming client connection. When a client connection is made, an instance of the socket for the new connection is created and returned by an "accept" method. The server can then communicate with the client using the returned socket's input stream and output stream.

The client-side program can be implemented in any programming language that allows TCP/IP sockets to be created (Java is used here to allow the TINI boards to be implemented). The TCP client first constructs a socket instance to establish a TCP connection to a specified remote host and port. It then communicates using the socket's I/O streams as in the case of the server.

In order to test the hybrid environment proposed previously, the benchmark manufacturing scenario proposed by Cavalieri et al. (2000) was simulated in Arena as shown in Figure 4. However, rather than simulating all of the machines in Arena, one of the machines is implemented by a TINI board: i.e., the communication interface for the TINI board, "ARENA_RSPR_TEST", is inset in Figure 4.

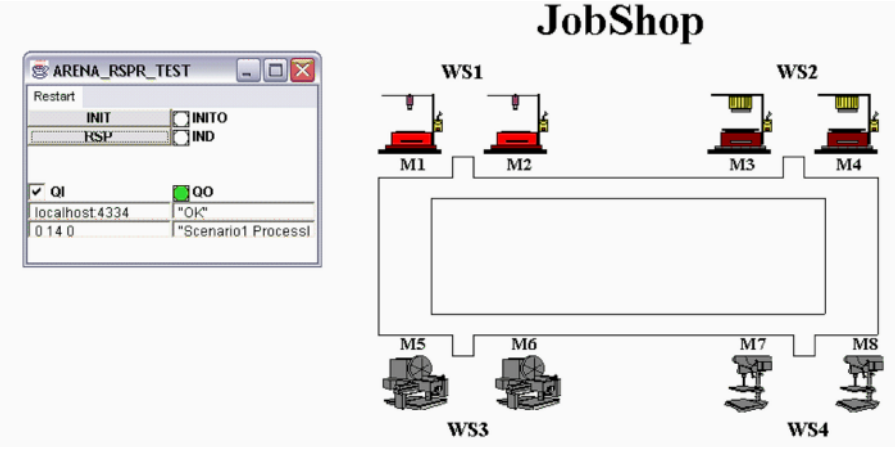


Fig. 4. A snapshot of the hybrid environment (simulation side)

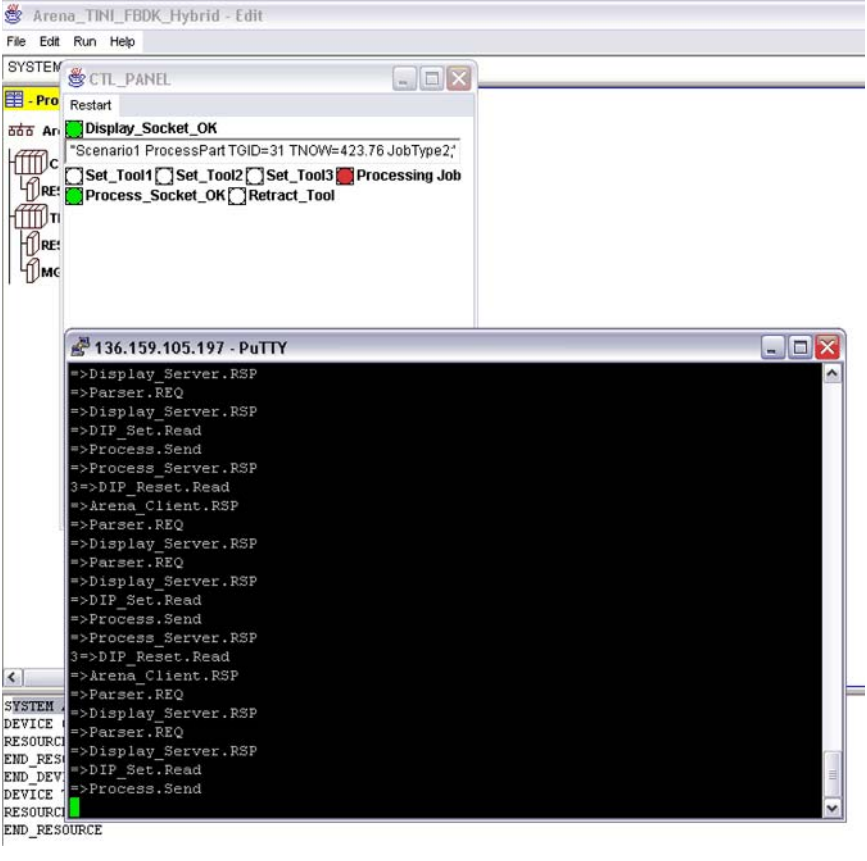


Fig. 5. A snapshot of the hybrid environment (physical side)

In order to implement the machine functionality on the TINI board, the control application was developed as an IEC 61499 function block application using Function Block Development Kit (FBDK) (Christensen, 2005). A screen shot of the physical device side of the hybrid environment is shown in Figure 5.

This figure shows the FBDK configuration screen in the background ("Arena_TINI_FBDK_Hybrid – Edit"), the TINI board control HMI in the upper left ("CTL_PANEL"), and a PuTTY terminal window connecting to the TINI board in the foreground ("136.159.105.197"). At the time of the screenshot, the TINI board was processing a part while the Arena model waited for a response. The processing time in this case is dependent on the status of the TINI board's discrete I/O.

4 Implementing the Client-Server

In this section we look at the general design of the client-server software for the hybrid environment. Since the problem of implementing client-server communications is common, we focus on developing a software design pattern for this problem. A design pattern is "a generalized solution to a commonly occurring problem" (Douglass, 1999). To be a pattern, the problem must recur often enough to be usefully generalizable. The solution must also be general enough to be applied in a wide set of application domains. For this paper, we develop an extension of the Proxy Pattern (Douglass, 1999), which abstracts the true server from the client by means of a "stand-in" or surrogate class providing a separation of a client and a server, allowing the hiding of specified properties of the server from the clients.

The proxy pattern adds a proxy between the Abstract Client and the Abstract Server. The pattern has two sides. In the first side, the Client-side Proxies subscribe to the Server-side Proxies, which publish the data under the command of the Concrete Servers. When the Concrete Servers call a send() operation, all the remote Client-side Proxies are notified of the new data. On the other side, the Concrete Clients subscribe in turn to the Client-side Proxies, just as in the Observer Pattern, where Concrete Clients subscribe to Concrete Servers. When these Client-side Proxies are notified of the new data, they send the data to all their local subscribers.

Although the structure of the pattern emphasizes the exchange of Data objects, this is only one kind of service that can be performed via the Proxy Pattern. In fact, any service may be published by the server and accessed via the proxy classes, even if no data is actually exchanged.

For the hybrid model a similar setup serves well for the processing and data exchange involved. In this case however, the data exchanged between the Arena model and the client application is that information specific to jobs to be processed in the embedded controller. The data exchanged is passed in the form of strings to a TINI function block application where the local proxy, namely the proxy function block will receive the information. This data/information is extracted by local Service Interface Function Blocks (SIFB's) to process the information according to the nature of the job.

4.1 The Client-Side Proxy

The Client-side Proxy is a specialized proxy that serves as the local "stand-in" for the remote server. Its clients are local, so it uses localized Notification Handles so that when it receives updated information from its associated Server-side Proxy, it can notify its local clients. It must unmarshall the data messages and reformat the Data Object into local format from network format. It subscribes to the Server-side Proxy that ultimately provides the marshaled data from the Abstract Server. The Client-side Proxy usually subscribes to the Server-side Proxy immediately upon its creation or as soon as its first client subscribes (Douglass, 1999).

The Client-side proxy in the hybrid system exists in the FBDK environment and is embedded in the TINI board. The functions of this proxy can be incorporated into a function block. The function block class encompasses the events, algorithms and data required to marshall and unmarshall the data messages to and from the Server-side Proxy. The Client-side proxy uses Java client socket communication features as described in (Brennan and Soundararajan, 2005).

This proxy's client is the function block application that performs the emulation of the job processing. In effect, this proxy immediately subscribes to the Server-side Proxy upon its creation establishing the socket communication. The subscribed message will be the job processing message that contains the necessary attributes in string format.

The marshaled message is then passed onto the function block application where the message is parsed to extract required job information. Upon completion of the job processing by the TINI board a "process done" event triggers the Client-side Proxy to marshall the Server-side proxy for subsequent data messages.

4.2 The Server-Side Proxy

The Server-side Proxy provides encapsulation of the Abstract Server from the communication media and protocols. It manages remote subscriptions from Client-side Proxy objects and notifies them when data is "pushed" to it by the Abstract Server. It is responsible for marshalling the information into a network or bus message and converting the data values into network format. The Server-side Proxy usually subscribes to the Abstract Subject immediately upon its creation (Douglass, 1999). The Server-side Proxy in the hybrid system is the DLL described previously, which is responsible for creating the server socket and managing requests of client through the TCP/IP protocol stack.

4.3 The Proxy Pattern

The hybrid system is unique since there is a simulation model and a real-time module. The simulation model and the inter process communication is setup as per the Arena modeling styles; the client-side application is to be implemented as per the IEC 61499 standard. Therefore, the proxy pattern implementation has to be fine tuned to the specific environment. It is not possible to have the class relations in the implementation exactly as specified in the default template of the pattern but nevertheless the implementation does adhere to the proxy pattern specification as illustrated in Figure 6.

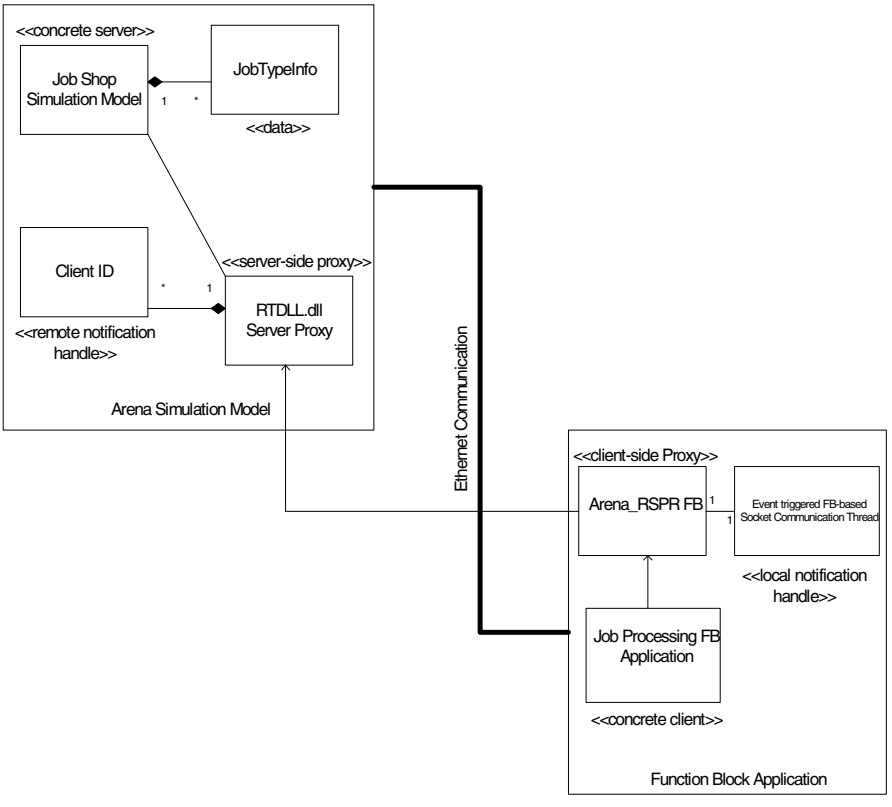


Fig. 6. The proxy design pattern for the hybrid environment

5 Discussion

The main objective of our research reported in this area is to be able to extend the work on benchmarks to the lower, real-time control level. Our current work on the hybrid environment is concerned with this aspect of the research. In particular, we are currently implementing physical agents or "holons" (McFarlane and Bussman, 2000) on the TINI platform that will interact with the hybrid environment in order to provide intelligence at the physical device level. These physical agents are implemented using IEC 61499 function blocks (IEC, 2000) as described in (Olsen et al., 2004), which allow our work on manufacturing scheduling and control (Brennan and O, 2004) to be extended to include real-time aspects.

In this paper, we have reported on one aspect of this research: the development of a Proxy Pattern to support client-server communication in the hybrid environment. This work has shown that the Proxy Pattern does a good job of isolating the subject from knowledge that the server may be remote. The advantage is that the clients are simplified, not having to deal differently with remote and local clients. The Proxy Pattern also encapsulates the knowledge of how to contact the servers into the proxy

classes so that should the communications media change, fewer classes must be updated (Douglas, 1999).

Because there are usually many fewer client-proxy instances (one per data type per address space) than client instances, the traffic on the communications media is minimized. One message is sent across the bus or network for each proxy, rather than one per client. This reduces bus traffic, a common bottleneck in embedded and real time systems. Bus traffic is reduced even further because of the use of a subscription policy, resulting in transmission of the data only when necessary, as opposed to polling for the data.

In the hybrid environment the pattern does a good job of isolating the subject from knowledge that the server may be remote. The client is simplified by not requiring the server proxy to deal differently with all the local function blocks to subscribe data from the server. The proxy also encapsulates the knowledge of how to contact the server and what communication process it requires to talk to the server. Thus if the communication media changes or if the client environment changes the proxy can be changed to suit the needs appropriately. Also if the client environments do not support a Java virtual machine, JNI's (Java Native Interfaces) can be built to serve the purpose.

References

Brennan, R.W., and Soundararajan, K. (2005) "A hybrid simulation/physical environment for benchmarking real-time distributed control systems". *Proceedings of the 16th IFAC World Congress*.

Brennan, R.W., and W. O (2004) "Performance analysis of a multi-agent scheduling and control system for manufacturing". *Production Planning and Control*, 15(2), pp. 225-235.

Cavalieri, S., M. Garetti, M. Macchi, and M. Taisch (2000) "An experimental benchmarking of two multi-agent architectures for production scheduling and control". *Computers in Industry*, 43(2), pp. 139-152.

Christensen, J.H. (2005) *HoloBloc Website*, http://www.holobloc.com.

Douglass, B. (1999) *Doing Hard Time: Developing Real-time Systems with UML, Objects, Frameworks, and Patterns*, Addison-Wesley.

IEC TC65/WG6 (2000) Voting Draft – Publicly Available Specification - Function Blocks for Industrial Process-measurement and Control Systems, Part 1-Architecture, International Electrotechnical Commission.

Kelton, W., R. Sadowski, and D. Sadowski (1998) *Simulation with Arena*, McGraw-Hill, New York.

Loomis, D. (2001) *The TINI Specification and Developer's Guide*, Pearson.

McFarlane, D.C. and Bussmann, S. (2000) "Developments in holonic production planning and control", *Production Planning and Control*, **11**(6), 522-536.

Olsen, S., J. Scarlett, R.W. Brennan, and D.H. Norrie (2004) "Contingencies-based reconfiguration of holonic control devices", *6th IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services*, Vienna, Austria.

Shen, W., D.H. Norrie, and J-P. Barthes (2001). *Multi-agent Systems for Concurrent Intelligent Design and Manufacturing*, Taylor & Francis.

Information Access and Control Operations in Multi-agent System Based Process Automation

Ilkka Seilonen[1], Teppo Pirttioja[2], Antti Pakonen[3], Pekka Appelqvist[2], Aarne Halme[2], and Kari Koskinen[1]

[1] Helsinki University of Technology, Computer and Information Systems in Automation, P.O.Box 5500, FIN-02015 HUT, Finland
{ilkka.seilonen, kari.o.koskinen}@hut.fi
http://www.automationit.hut.fi
[2] Helsinki University of Technology, Automation Technology Laboratory, P.O.Box 5500, FIN-02015 HUT, Finland
{teppo.pirttioja, pekka.appelqvist, aarne.halme}@hut.fi
[3] VTT Industrial Systems, P.O.Box 1301, FIN-02044 VTT, Finland
antti.pakonen@vtt.fi

Abstract. An approach to combine information access and control operations in a process automation system extended with multi-agent system technology is presented in this paper. According to this approach a multi-agent system supervises an ordinary process automation system by performing higher-level information access and control operations. The information access operations are aimed for actively combining information from different sources depending on the monitoring tasks of the users. The control operations of the multi-agent system are supervisory control tasks performed either in sequential or iterative fashion. The expected benefit of the multi-agent system is enhanced adaptability of the automation system and increased situation awareness of its users. The architecture of the multi-agent system is based on the BDI agent model and utilization of so-called ontologies. An approach for engineering applications for this kind of a multi-agent system is also discussed. The approach is demonstrated with results from experiments performed with industrial test data and a laboratory test process.

1 Introduction

This study is motivated by the expected utility of multi-agent systems (acronym: MAS) applications in process automation and some limitations of the earlier research in this field. The primary utility of MAS in process automation has been assumed to be adaptation to various change situations. Several research efforts applying MAS to process automation primarily with this motivation have been reported. Most of this research has focused on the control functions of process automation. There has been limited attention to other functions, e.g. monitoring. However, studying the application of MAS to a broader set of functions of process automation than just control could be useful.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 144 – 153, 2005.

The purpose of this paper is to present an approach for combining information access and control operations in a MAS-based extension of a process automation system. According to this approach both information access and control operations are designed into the same MAS. The paper is outlined as follows. Chapter 2 will discuss MAS applications in process automation. The approach for MAS based information access and control operations in process automation is presented in Chapter 3. Some results from experiments are depicted in Chapter 4 followed by conclusions in Chapter 5.

2 Multi-agent Systems and Process Automation

Research about MAS applications in process automation has been less extensive than in discrete manufacturing. A possible reason for this is that both the suitability and usefulness of MAS in process automation is maybe not so evident than in discrete manufacturing [2].

In the current research of MAS applications to process automation agents have been used for a few different purposes. Agents have been proposed as modules of an automation system [15], an integration mechanism [3] and a new type of intelligent controllers. These MAS-based controllers have been applied to continuous [1][17], sequential [9][16] and batch control [8]. The Contract Net and market-based negotiation have been used as a coordination mechanism. Internal agent models have been based on rule-based reasoning or the BDI-model and planning. The approaches have been different depending on the type of control. A complete set of cooperation mechanisms and associated agent model applicable for several types of control operations have not been presented.

Application of MAS to the information access operations of process automation may be argued to be justified. MAS are assumed to be useful particularly in dynamic environments with heterogeneous and distributed data sources [4][6][7][10]. In this situation also the usage of ontologies in agent communication is expected to enhance the accessability of different systems [11]. Recent developments in this field, e.g. OWL [12] could be useful in expressing and fusing knowledge also in process automation. This kind of sophisticated system integration methods are one possibility for handling the increasing amount of electronic information is process automation.

The approach presented in this paper is aimed for overcoming the fragmented state of the current research in MAS applications in process automation. In order to do this, a MAS architecture and programming model for both control and information access operations in process automation is presented. The model covers both control and information access operations. In the long run the presented approach aims for a specification of a generic MAS for process automation.

3 Multi-agent System Based Process Automation

3.1 Extended Automation Architecture

According to our approach an extended process automation system consists of two layers as illustrated in Fig. 1. A MAS supervises an ordinary process automation

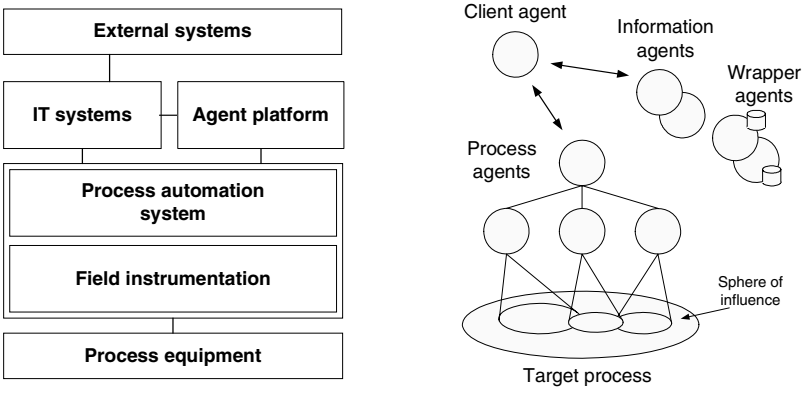


Fig. 1. Architecture of a process automation system with a MAS extension and an example of society of information and process control agents

system. The purpose of the MAS is to monitor the lower-level automation system, deliver enhanced information about its operation to external users and reconfigure the operation logic of the automation system when needed.

The MAS extending the process automation system consists of two types of agents: process agents and information agents. Both types of agents communicate according to the specifications of the FIPA-standard [5]. The process 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. Furthermore, information agents cooperate with process agents to perform advanced monitoring tasks.

The process agents cooperate via both vertical and horizontal channels. The purpose of the vertical cooperation is to decompose information access or control operations into parts, e.g. operations relating to a certain subprocess. The horizontal cooperation 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 process agent combines cooperation on both of the cooperation directions.

3.2 Control Operations

The process automation agents perform supervisory control operations either in sequential or in iterative fashion. The former may relate e.g. to process state change or batch control operations and the latter to tuning of continuous control. In the sequential control case the agents have a process state as a common goal, to which the process should be taken. The task of the agents is first to plan a shared sequence of control actions and then execute this sequence. In this case the operation of the agents can be characterized as distributed planning. In the iterative control case the agents have conflicting goals, which they have to balance. The task of the agents is to search for optimal values for their supervisory control variables. In this case the operation of the agents can be characterized as distributed iterative search.

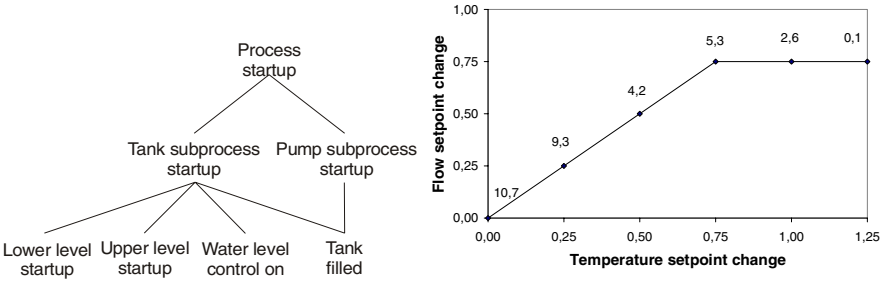


Fig. 2. Examples of different search spaces in distributed planning of control sequences and iterative refinement of supervisory control variables

The operation of the process automation agent society shares some common features in both types of control operations. In both cases the agent society executes a distributed search process, in which cooperation among the agents is performed via negotiation. The search spaces are decomposed according to the structure of the controlled process. The differences between the two types of search spaces appear in the form of goals and the search processes. In the sequential control case the goals represent known states, to which the process should be taken. The search process is performed in a backward chaining fashion. In the iterative control case the goal is a function of process variables and the optimal values are not known beforehand. The search process is performed in a forward chaining fashion and guided with feedback from the process during iterations. The search processes are illustrated in Fig. 2 and described in more detail in other publications [15].

The BDI-agent model of the process automation agents may be used for both of the control operation types. However, depending on the control type the internal agent operation appears in a different form. In the distributed planning of control sequences the local planning processes of all the agents are similar parts of the cooperative search process. The input to the local planning tasks of the agents consists of the requested goals and current values of process measurements. The outcome from the planning tasks of the agents is plans representing agent control sequences and contracts representing their coordination with other agents. In the iterative refinement of supervisory control variables the local decision-making processes of the agents are different depending on their role. The input to the local decision-making activity of a coordinator agent consists of the global control objective and the current values of process measurements. The input to the local decision-making activity of participant agents consists of the goal received from the coordinator and current values of process measurements. The outcome from the decision-making tasks of the agents is new values of control variables.

3.3 Information Access Operations

The information agents perform information search or monitoring operations in a distributed manner. Their operation has its foundation in combining information from field device measurements, operational state classification, simulations, condition monitoring and process models. All of these subsystems produce partial information

about the performance of the monitored process, but typically their representations are incompatible. In here, the information agents can utilize ontology-based knowledge representation to overcome this problem. Furthermore, the overall information access procedure is based on goal-oriented operation, where goals are used to represent information needs (see Fig 3). When an information access request is received, an agent finds out a sequence of operations that it has to perform to generate the result. The task of the agent is to decide what partial information it needs, find out sources for them and infer a conclusion from them. This way of performing the information access operations resembles the planning control operations. In the case of a failure to obtain the requested information, an alternative operation sequence is performed if one is available. The operation sequences may also change if needed.

The operation of the information agent society is based on distributed search and processing of information. Various agent negotiations are used to gain cooperative and dynamic operation. The search of information is decomposed using the understanding of physical structure of the monitored process, its present state, various diagnostics reports and abilities of different information providers. Basically the idea is to utilize the proactive interaction mechanisms to adapt to changes in the information, the environment, or the preferences of the users. In addition to this, searching agents use ontology-based knowledge representation based on domain ontology when they negotiate about the information goals. The selection of appropriate agent interaction protocols for certain monitoring tasks is discussed further in other publications [13].

The operation of information agents is based on the same BDI-agent model, where information access operations are seen as means to achieve information goals. The goals itself may be decomposed to subgoals, which may be processed locally or may require access to information of other agents. Processing of an information goal may initiate a few different types of information processing operations. Currently, we have divided these operations to three subclasses. Integration operations contain searching and transforming data, which is already available somewhere in the system.

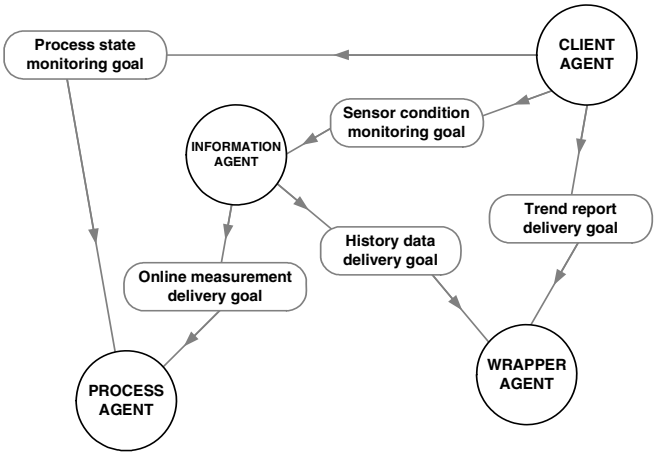


Fig. 3. An example of information access related goals shared between information agents

Monitoring operations are active watchdogs, which are needed for successfully supervising changes in some part of the process related data. Reasoning operations are used to produce derived information about the process performance starting from the basic monitoring data and its modeled relationships. Results from the early experiments with information access requires study further the issue.

3.4 Engineering

Applications for both control and information access operations are intended to be designed in a similar way and partially within the same agents. In both cases the applications conform to the extended automation system architecture, which separates an agent platform for process automation from actual applications. The process agents may be used for both control and information access applications while the information agents are intended only for information access applications.

The application design process is proposed to proceed in three phases, which may interleave each other. The first phase is the design of the agent society by identifying necessary agents and their intended interactions. The interactions as assumed to be specified mainly as goals, which the agents exchange. The second phase is the design of external connections of various agents. These can include I/O to the controlled process and connections to external data sources. The third phase is the design of the internal logic of agents. This part of the design contains several aspects. A major part of the application logic of agents is to be defined with plans, which describe how agents are going to fulfill their goals. In the plans the agents utilize models and modules. The primary target of modeling is assumed to be the controlled process, which may be modeled e.g. with the techniques of qualitative process modeling. The modules may wrap numerical algorithms for control calculations information processing. The plans, models and modules of an agent are intended to be glued together with a set of ontologies, with which these are expected to be compliant. However, the ontologies needed for this purpose are currently in very early stages of their development.

4 Experiments

4.1 Control Experiments

The control operations of the process agents have been demonstrated with a laboratory test environment. The test environment consists of a test process, an automation system and a prototype agent system implementation with five agents [15]. The controlled process in the test environment is a small-scale water temperature control process. The control system is contains three control loops: one for water level control and two for water temperature control at upper and lower levels of the tank. The objective of the water temperature control is to keep the temperature at given values at both levels of the tank.

Two different test scenarios have been used for experiments with the test environment. The deliberative type of control operations has been demonstrated with startup sequence planning and the iterative type of operations with a fault recovery scenario [15]. The objective of the agents in the startup scenario is to run the test process into a

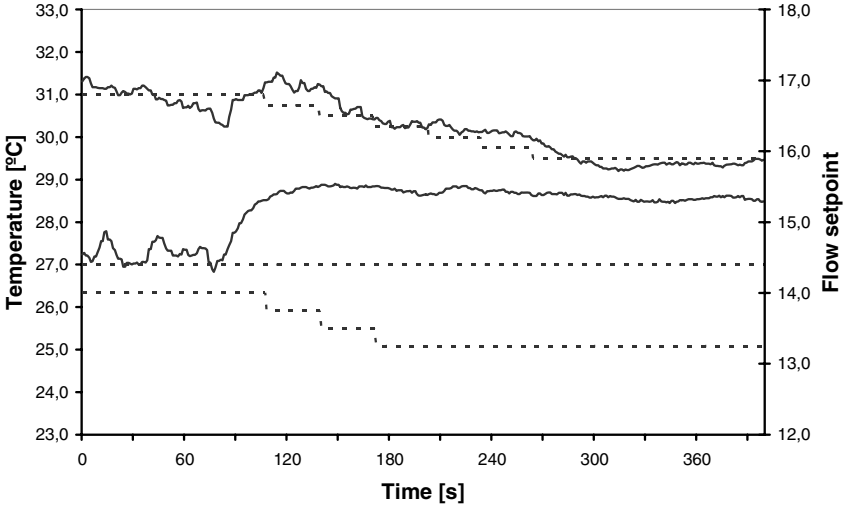


Fig. 4. Process measurements and setpoints during a test run of the fault recovery scenario. The solid lines represent temperature measurements. The dotted lines represent set-points of the temperatures and the water flow.

normal operation state. The state change is carried out with execution of a sequence of control actions to be planned by the agents. The agents have to adjust the startup sequence for different initial states, e.g. a full tank does not need to be filled. The objective of the agents in the fault recovery scenario is to compensate the effects of a fault. The fault situation is a simulation of failure of one of the control valves in the system. Before the fault there is a proper temperature profile in the tank. After the fault the profile can not be fully maintained. However, the effect of the fault can be partly compensated via changing the setpoints of the other controllers in the system.

Results from the experiments seem to support the feasibility of the approach for control operations in a laboratory environment. The results of the fault recovery scenario are illustrated in Fig 4. In this scenario three subprocess agents react to a fault situation with the iterative control scheme. First, one of the agents detects the fault and makes an inference that even it can compensate the fault partially with its local recovery capabilities it cannot control its control variable properly any more. It initiates negotiation for help with the other two agents. Both of these offer to change their setpoints in order to reduce the problem. The first agent accepts both proposals. The negotiation process is then repeated iteratively with new setpoints at each step until the control errors of the controlled temperatures are balanced.

4.2 Information Access Experiments

The information access operations of the process agents have been demonstrated with a setup where real process data is analyzed off-line. The tests are related to condition and process monitoring functions in bleaching of mechanical pulp. In this process the important process variables of a plant are pulp brightness, production rate, pH, and chemical residual. In addition to online sensor measurements, product quality and

process conditions may be monitored with laboratory measurements. In the bleaching process all physical equipments are exposed to mechanical vibrations, corrosive chemicals and wide temperature variations, as is the case in many other process automation setups too. Therefore, to be able to assure the correct operation the potential breakdowns of instrumentation must be actively and effectively monitored in all times.

The objective of these tests has been to show that the MAS based architecture using goal-oriented operation is feasible in real monitoring functions. There have been three information access test experiments so far. First of these was related to fusing of process monitoring information retrieved from different data sources. In this scenario the main contribution was to dynamically search agents, which that provide data from legacy databases and then convert this data into shared ontology [13].

In the second test scenario the objective was to aid human user to verify correct operation of physical measurement, namely the pH-transmitter. In this scenario, the unreliable value of an online measurement device is compared to reliable laboratory measurement on certain time intervals. In practice the operation is not currently automated because the separate information is in different formats and locations and the laboratory measurements are available on an irregular basis.

The third experiment was about process state classification using hierarchical MAS organization. The motivation for this test was the observation that many of present day monitoring algorithms assume a normal operation mode of the monitored process and therefore fail to make correct analysis in abnormal conditions. The approach used simple rule based operation scheme in the instrumentation level and produced short ontology based descriptions about the process state, which would be used by other agents. In the upper parts of the agent organization the analysis of lower level subprocess states were combined to produce a conclusion about the whole plant.

Results from the experiments are fairly preliminary, but on their basis one can argue that the information access operations may be build based on the proposed architecture. Although agent based information access in general is an active research area, its applicability to the information access tasks of automation is not widely tested. There is not so much experience in designing this kind of applications and there are no generally available domain ontologies for automation. Therefore, the experiments were started with simple examples where the basic operation principles are tested. In the future more complicated test scenarios are needed to demonstrate the usability of the MAS based information access operations in process automation.

5 Conclusions

In this paper an approach to combine information access and control operations in a process automation system with MAS technology is presented. The approach defines architecture of an extended process automation, in which MAS supervises ordinary automation system. Both supervisory control operations and information access operation can be implemented with the MAS. The approach is complemented with an application engineering scheme utilizing the BDI agent model and ontologies. In the long run the approach aims for a specification of a generic MAS platform for process automation, with which one could implement several different kinds of cooperative and intelligent supervisory applications.

Although the current studies with the presented approach seem to suggest its initial feasibility, there are several open questions remain as subjects for further research. The properties, e.g. completeness, complexity and optimality of the search processes used in the control operations are not understood well enough. The information access operations studied so far are rather limited. The applicability of the approach to more useful information access scenarios has not yet been tested. The usage of ontologies is still in rather early stages. A final common open issue for both of the functionalities is their scalability to tasks involving a larger number of agents and more complicated cooperative processes.

References

1. van Breemen, A., de Vries, T. : An agent-based framework for designing multi-controller systems. In: Proceedings of the 5th International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, Manchester, UK (2000) 219-235.
2. 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
3. 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)
4. Ferber, J: Multi-agent systems: An introduction to distributed artificial intelligence. Addison-Wesley (1999)
5. FIPA: http://www.fipa.org
6. Jennings, N. R.: An agent-based approach for building complex software systems. Communications of the ACM, Vol. 44, No. 4. 35-41
7. Klusch M.: Information agent technology for the Internet: A survey. Data & Knowledge Engineering, Elsevier, Vol. 36 (2001) 61-74
8. Kuikka. S.: A batch process management framework, Domain-specific, design pattern and software component based approach. VTT Publications No. 398. Espoo, Finland (1999)
9. 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)
10. Nodine, M., Ngu, A., Cassandra, A., Bohrer, W.: Scalable Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth™. IEEE Transactions on Knowledge and Data Engineer-ing, Vol. 15, No. 5 (2003)
11. Obitko, M. Marík, V.: Adding OWL semantics to ontologies, In: Proceedings of the 1st International Conference on Applications of Holonic and Multi-Agent Systems (HoloMAS 2003), Prague Czech Republic (2003) 189-200
12. OWL.: Web Ontology Language, http://www.w3.org/TR/owl-ref/
13. Pirttioja, T, Seilonen, I., Appelqvist, P., Halme, A., Koskinen, K.: Agent-based architecture for information handling in automation systems. In: Proceedings of the 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS 2004). Vienna, Austria (2004)
14. Sanz, R.: Agents for complex control systems. In: Samad, T., Weyrauch, J. (eds.): Automation, control and complexity. Wiley & Sons, England (2000) 171-190

15. Seilonen, I., Pirttioja, T., Appelqvist, P., Halme, A., Koskinen, K.: Modelling cooperative control in process automation with multi-agent systems, In: Proceedings of the 2nd IEEE International Conference on Industrial Informatics (INDIN 2004). Berlin, Germany (2004)
16. 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
17. Ygge, F., Akkermans, H.: Decentralized markets versus central control: A comparative study, Jornal of Artificial Intelligence Research, Vol. 11. (1999) 301-333

An Initial Automation Object Repository for OOONEIDA

R.W. Brennan

Department of Mechanical and Manufacturing Engineering,
University of Calgary, 2500 University Drive NW,
Calgary, Alberta T2N 1N4, Canada
rbrennan@ucalgary.ca

Abstract. This paper focuses on the recent work on the Automation Object repository (AORepository) for the Open, Object-Oriented kNowledge Economy for Intelligent inDustrial Automation (OOONEIDA). The AORepository is currently based on the University of Calgary's CAREO (Campus Alberta Repository of Educational Objects) architecture and is supported by the CANARIE CA-NET4 network. In this paper, we describe the general architecture of the AORepository and provide examples of its use in the OOONEIDA initiative.

1 Introduction

Efficient handling of embedded intelligence is of central importance to the realization of agile production systems. To address this need, a new IMS Community of Common Interest (CCI) initiative (CCI, 2005) called OOONEIDA, or the Open, Object-Oriented kNowledge Economy for Intelligent inDustrial Automation, was initiated to enable the flexible, open integration and reconfiguration of embedded intelligence in industrial automation systems. The goal of OOONEIDA is the creation of the technological infrastructure for a new, open knowledge economy for automation components, products and systems. It is intended that this new economy will bring tangible benefits for all players in the value creation chain: device vendors, machine vendors (OEMs), system integrators and industrial enterprises will be able to encapsulate their intellectual property (IP) into re-useable, portable software components (automation objects) and to deploy these components into intelligent devices, machines, systems and automated factories.

To accomplish this goal, OOONEIDA is organized as a Canadian not-for-profit corporation linked to a network for the pursuit of industrial automation objects research, standards development, and training at each stage of the value added chain. In particular, OOONEIDA focuses on the development of open standards to foster the dissemination of already developed industrial automation technologies.

AT the heart of OOONEIDA lies its repository of public domain, for sale, industrial automation devices, machines, systems, tools, software and platforms, etc. The marketplace component of the repository will be an Internet enabled, world wide

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 154 – 164, 2005.

marketplace for accessing the tools, platforms, software and intelligent devices, machines and systems necessary to develop and implement the next generation of industrial automation systems within and among distributed enterprises.

This paper focuses on the initial work on the OOONEIDA Automation Object Repository (AORepository) performed at the University of Calgary. We begin with background on the general goals, technology and business drivers behind OOONEIDA. Next, we provide an overview of the AORepository in section 3 followed by a description of its general architecture in section 4. In order to manage the AORepository, an open metadata management tool is currently being used, which is described in section 5. Finally, we conclude the paper with a brief summary of the next steps in the development of the AORepository.

2 The OOONEIDA Community of Common Interest

OOONEIDA emerged from a growing consensus among researchers and firms operating in the industrial automation field that realized that new ontologies and open standards for industrial automation objects are necessary for effective and efficient systems integration at various levels in the industrial automation value chain up to and including enterprise integration.

A second driver for OOONEIDA is the expressed desire of IMS project participants in completed projects that focused on industrial automation (HMS (HMS, 2002), PABADIS (PABADIS, 2005), etc.) to foster actively the dissemination of the R&D results of these projects. In this respect, OOONEIDA is about the dissemination of Holonic Manufacturing Systems (HMS) technologies in plants and across extended enterprises often employing distributed information systems (e.g., PABADIS). It is also about the ontologies underpinning enterprise integration, concepts explored and developed in the NGMS IMS project as well as in Globeman 21 and Globemen.

The global market for automation products can be thought of in terms of a value creation chain. As illustrated in Figure 1, very similar activities are performed at all levels of the value creation chain. Device vendors design and produce parts of machines (i.e., mechatronic components) using basic components such as sensors, actuators, embedded computing devices, and system software: they add their own expertise in devices and partially encapsulate it in software components.

At the next level, machine vendors design functionally complete production machines using the component devices and develop software components for control, communication, visualization, etc. At the level of the system integrator, customer needs are fulfilled using the machines and components from the lower levels: they encapsulate their integration expertise in the form of software components which make the systems run.

In summary, all players in the automation value creation chain illustrated in Figure 1 deal with the integration of knowledge and its encapsulation for future use and re-use. Each of these players should also benefit from the unification and standardization of their components, and in particular, software components where design solutions for different configurations of products could be used without customization for each vendor requirement. As a result, the main goal of OOONEIDA is the creation of the technological infrastructure to integrate each level of the value creation chain.

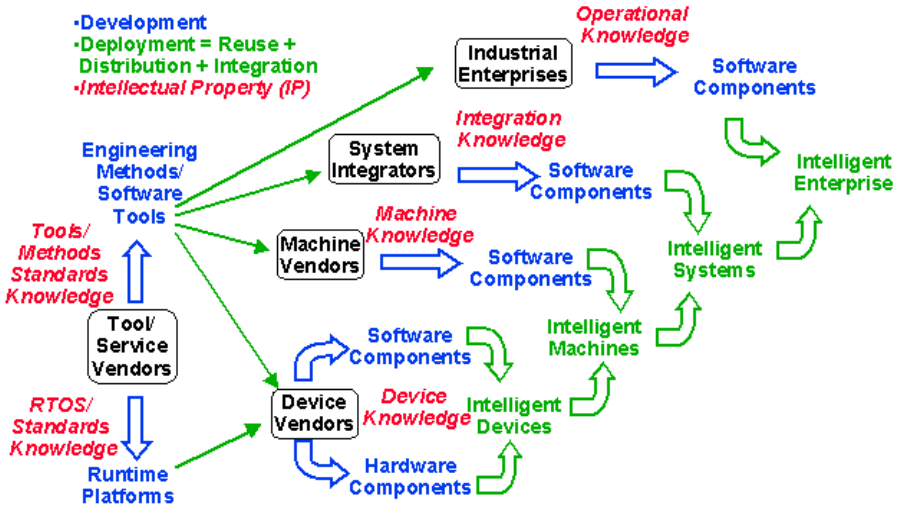


Fig. 1. Industrial automation value creation chain

3 The OOONEIDA AORepository

In this section, we provide an overview of the open repository of Automation Objects, or AORepository, that will be populated by members of OOONEIDA. The AORepository is based on:

- ongoing, worldwide standardization activities in organizations following in the footsteps of International Electrotechnical Commission (IEC), International Standards Organization (ISO), W3C Consortium and Foundation for Intelligent Physical Agents (FIPA);
- results of research successfully performed in IMS, regional and national projects
- independent development activities of the CCI members.

As illustrated in Figure 2, a key ingredient in supporting the knowledge economy is the role of broker agents to enable all players in the value-adding chain to locate and retrieve the required IP on which to base their development activities, and to deposit results of their activities in searchable form accessible to other players.

OOONEIDA will demonstrate an architecture-centric approach to industrial embedded systems design, based on open global standards for embedded software and open toolkits. This will enable the time- and cost-effective specification, design, validation, realization and deployment of intelligent mechatronic devices in distributed industrial automation and control systems in both discrete manufacturing and continuous process industries.

This architecture will use software agents to enable users to locate, acquire and use the intellectual property needed to perform their value-adding activities and make the results available (possibly for sale) to other players in the knowledge economy. In particular, the AORepository will consist of distributed repositories for standards,

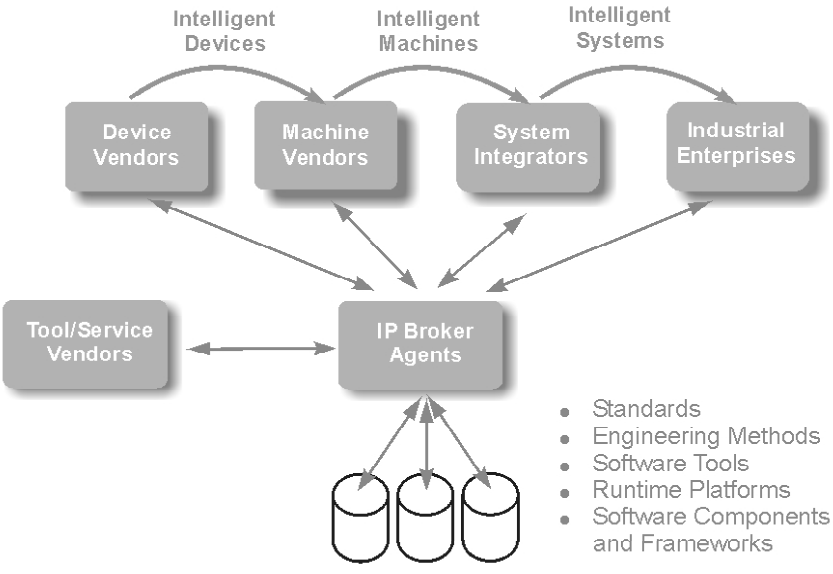


Fig. 2. Repository of automation intelligence in the automation value-creation chain

engineering methodologies and their supporting software tools, runtime platform implementations and software components. Systematic indexing and searching of these databases will be supported through appropriate adaptation of a proposed schema for "Automation Objects". In the following subsections, further details are provided on the current the implementation of the AORepository that is being developed at the University of Calgary.

4 The AORepository Architecture

The initial version of the OOONEIDA AORepository is based on the University of Calgary's CAREO (Campus Alberta Repository of Educational Objects) architecture (CAREO, 2005). The CAREO architecture was developed by Netera Alliance (Netera, 2005) and is supported by the CANARIE CA-NET4 network (CANARIE, 2005).

As illustrated in Figure 3, the repository architecture is based on a three-tier, client server model consisting of three main components: (i) the users, (ii) the repository application, and (iii) the metadata store.

Users of the Automation Object repository include device vendors, machine vendors, system integrators, industrial enterprises, and tool service vendors as shown in Figure 1 as well as repository administrators. These users only require a standards-compliant web browser (with streaming media protocols) to access metadata or publish and request distributed Automation Objects. The repository administrators would consist of various subgroups responsible for repository management (e.g., quality control for metadata and Automation Objects).

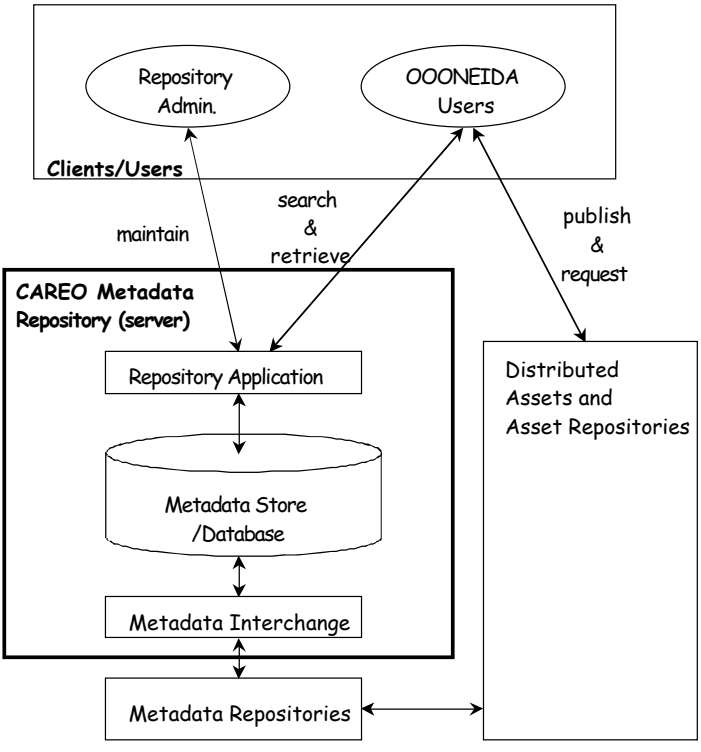


Fig. 3. The AORepository architecture

The repository application is the middle tier in the CAREO three-tier architecture. Netera Alliance has developed application software that provides search and retrieval access for clients/users to the metadata store that, to the user, appears as an HTML interface that can be used for searching and retrieving Automation Objects.

Currently, the repository is populated with IEC 61499-based (IEC, 2000) automation objects that have been developed using Function Block Development Kit (Holoboc, 2005). More specifically, the Odo Struger Laboaratory at the Technical University of Vienna has provided examples of mechantronic objects that integrate into a material handling system; as well, Profactor/FH-Wels has provided documented examples of IEC 61499 service interfaces for various kinds of I/O. Figures 4 and 5 show an example of a search and retrieval of an automation object respectively.

As can be seen in Figure 4, the user has various options available for each retrieved Automation Object. For example, users may retrieve the object (View), inspect its metadata (Details), or provide comments on the object (Discuss). In this case, the Automation Object is presented in an HTML format with an embedded Java applet. However, the CAREO repository is capable of supporting a wide range of media (e.g., Flash, Quicktime Video, etc.).

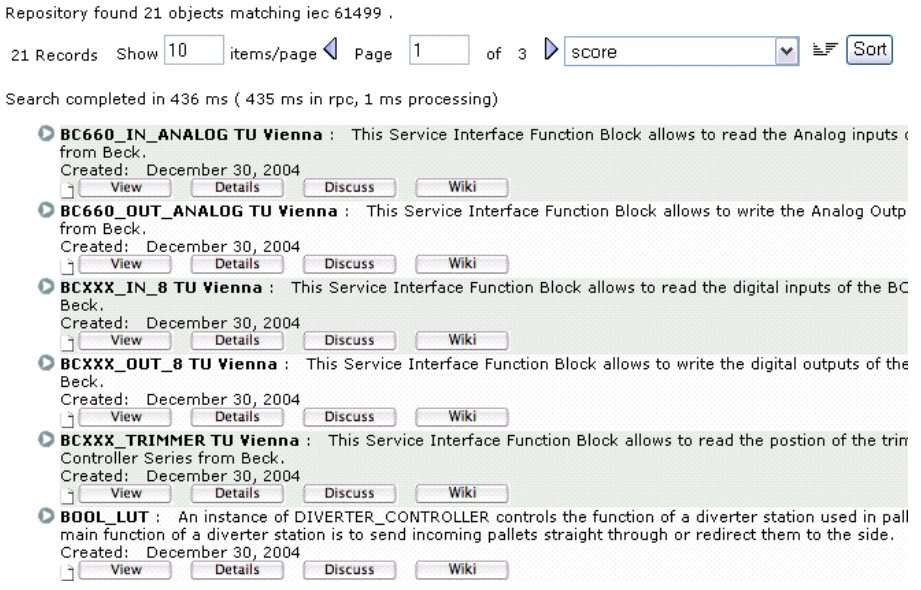


Fig. 4. A typical AORepository search

Finally, the metadata store (database tier) is the bottom tier of the CAREO architecture. Currently, the metadata is structured according to the CanCore Learning Object Metadata Protocol (CanCore, 2005), which is a subset of the IMS schema (IMS, 2005). Although this schema was developed for educational applications, it provides a very good starting point for an Automation Object schema. However, given the current work on using XML (eXtensible Markup Language) (W3C, 2001) to describe Automation Objects (e.g., IEC 61499 (IEC, 2000)), future work by the OOONEIDA community will focus on Automation Object specific schema for use in the database tier.

5 Managing the Automation Objects

The discussion to this point has primarily focused on the user interactions with the metadata server. Of course, a key requirement of the AORepository is the ability to allow the OOONEIDA community to populate repositories of Automation Objects. As shown in Figure 2, Automation Objects may be distributed on servers across the internet (i.e., “distributed assets and asset repositories”). For example, an Automation Object created by a tool/service vendor may be located on the vendor’s web server. In this case, metadata describing the Automation Object, or “asset”, would either be submitted directly to the CAREO repository by the vendor, or collected by CAREO via another repository or metadata store.

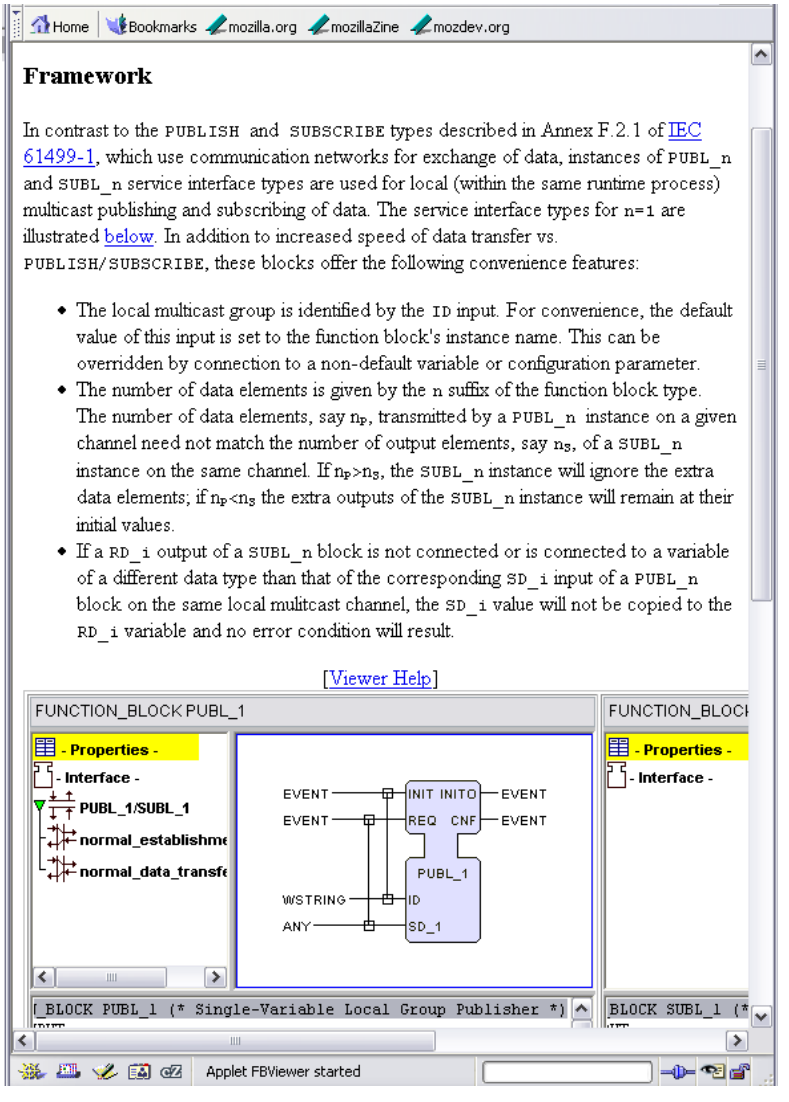


Fig. 5. A Typical AORepository Retrieval

In order to manage Automation Objects (e.g., load the object and the object's metadata into the repository) the CAREO web interface may be used. However, the current version of the CAREO web interface provides limited support for editing objects and their associated metadata. For example, users may view the most commonly used metadata fields, and if they have administrator permission, they can edit these fields.

For more advanced support of Automation Object indexing, loading and searching, the CAREO repository can interface with specialized application programming

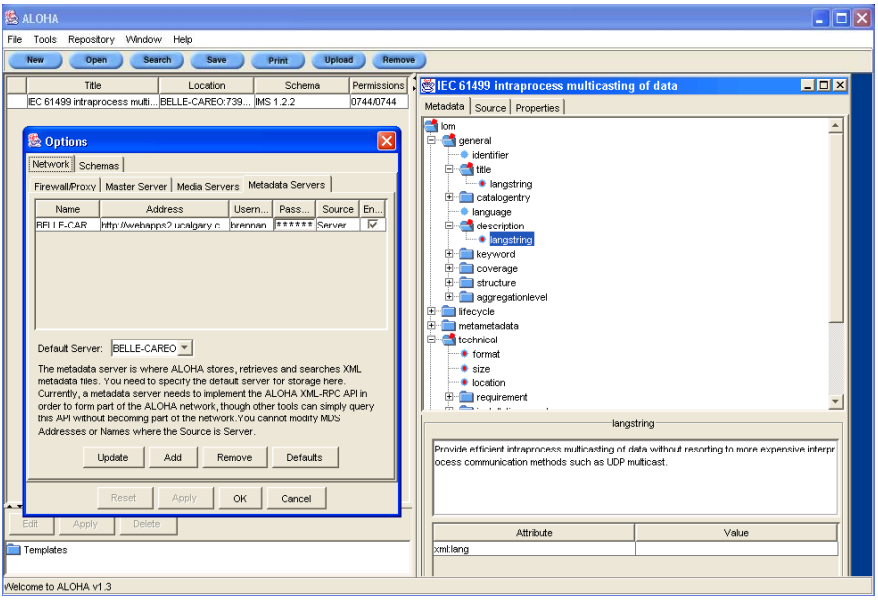


Fig. 6. Using ALOHA to manage the AORepository

interfaces (APIs) that support XML-RPC (Remote Procedure Call). For example, the Advanced Learning Object Hub Application (ALOHA, 2005) software tool, shown in Figure 6 is currently being used.

ALOHA serves as a client for the World Wide Web that can be plugged into multiple repositories and provides all of the features of the Automation Object repository portal shown in Figure 2, however its focus is on repository management. For example, users and administrators may specify the repositories to be used, assign access permissions, and load/edit Automation Objects. When a new Automation Object is loaded into ALOHA, it automatically selects the most appropriate media server.

As noted previously, the CanCore/IMS schema is currently being used, however work is being done on developing Automation Object specific schema. In Figure 6, a hierarchical view of the metadata for an Automation Object ("IEC 61499 intraprocess multicasting of data") is provided in the right window. Details of the metadata server that stores this object are provided in the "Options" window.

The key to indexing and retrieval in the CAREO system is the metadata store. For example, Figure 7 shows the full metadata hierarchy for the example given in Figure 6. In this case, the object's author can enter general information about the object such as its title, description, keywords, etc. that can be used for searching. As well, more technical information on the file type, its installation requirements, platform requirements, etc. can be provided to assist the user of the object.

As noted previously, XML is used to represent the metadata for each of the objects in the CAREO repository. For example, Figure 8 shows the XML code for the example described previously. It should be noted however that the ALOHA tool

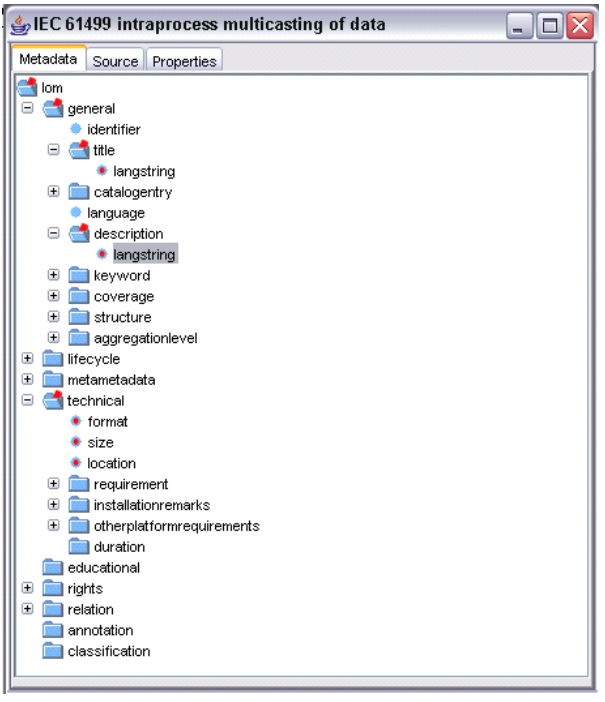


Fig. 7. The ALOHA metadata

supports any valid XML schema that can be used for indexing. For the current implementation of the OOONEIDA AORepository, the CanCore/IMS schema is being used. However, given that the majority of the automation objects in the AORepository will be based on the IEC 61499 standard, we are investigating using the Document Type Declarations (DTDs) for IEC 61499 data types, library elements, and function block management (IEC, 2000) as the metadata schema.

6 Next Steps

Currently, the ALOHA tool has proven quite useful for preparing Automation Objects for the repository. In particular, once the object has been developed, the tool allows the user to import the object into ALOHA for metadata editing and subsequent uploading to the repository. In order to develop an OOONEIDA specific Automation Object tool however, we are currently focusing our work in two main areas: (i) Automation Object schema development, and (ii) Automation Object editor development.

As noted previously, the first area focuses on the development of an Automation Object specific XML W3C Schema (W3C, 2001). As a starting point, we are looking at the Document Type Declarations (DTDs) for IEC 61499 data types, library

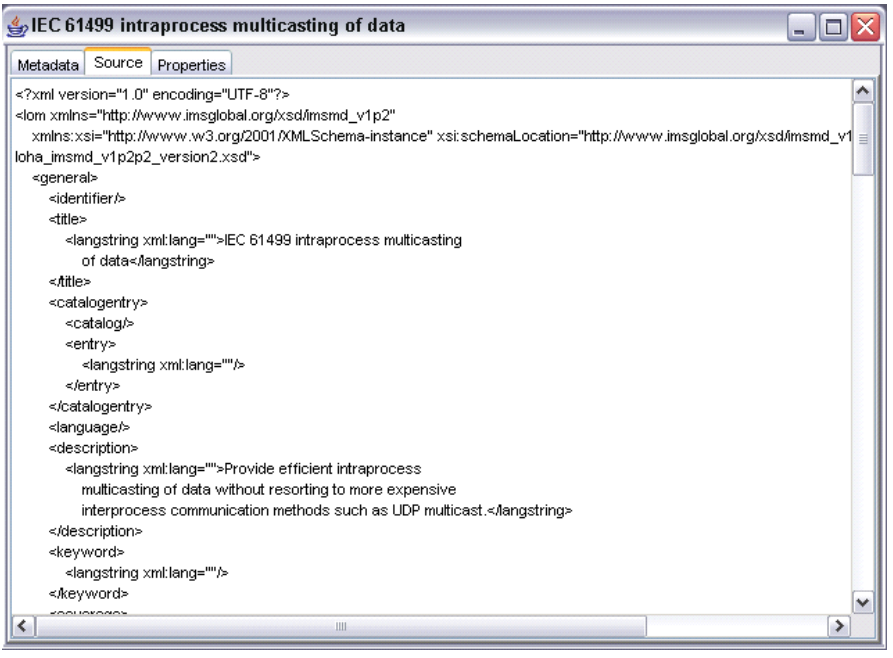


Fig. 8. The ALOHA metadata in XML

elements, and function block management (IEC, 2000). W3C XML Schema have been chosen over the existing DTDs primarily because they provide the extensive metadata capabilities required for accurate indexing of Automation Objects.

Given a clearly defined Automation Object "vocabulary" based on an Automation Object schema, Automation Objects can be represented by well-formed XML documents. These will then be used as input to an Automation Object Editor (AOEditor) that will be used to build Automation Object files and their associated metadata. In other words, the AOEditor will build on the ALOHA tool by automating the Automation Object generation and metadata generation tasks.

References

Advanced Learning Object Hub Application (2005) *Web Site*, http://aloha.netera.ca/.
Automation Object Repository (2005) *Web Portal*, http://careo.ucalgary.ca/oooneida/.
Bean, J. (2003) *XML for Data Architects*, Morgan Kaufmann Publishers.
Campus Alberta Repository of Educational Objects (2005) *Web Site*, http://careo.ucalgary.ca/.
Canadian Core Learning Resource Metadata Application Profile (2005) *Web Site*, http://www.cancore.ca/.
CANARIE (2005) *Web Site*, http://www.canarie.ca/.
Community of Common Interest (2005) *Web Site*, http://www.ims.org/index_cci.html.
Holbloc (2005) *Web Site*, http://www.holobloc.com.

HMS - Holonic Manufacutring Systems Consortium (2002) *Holonic manufacturing systems overview*, Holonic Manufacturing Consortium Web Site, http://hms.ifw.unihannover.de.

IEC TC65/WG6 (2000) *Voting Draft – Publicly Available Specification - Function Blocks for Industrial Process-measurement and Control Systems*, *Part 1-Architecture*, International Electrotechnical Commission.

IMS Learning Resource Meta-data Specification (2005) *Web Site*, http://www.imsglobal.org/metadata/.

Netera Alliance (2005) *Web Site*, http://www.netera.ca/.

PABADIS (2005) *Web Site*, http://www.pabadis.org/.

World Wide Web Consortium (W3C) (2001) *XML Schemas, W3C Recommendation*, (available at http://www.w3.org/)

Towards Engineering Methods for Reconfiguration of Distributed Real-Time Control Systems Based on the Reference Model of IEC 61499

Thomas Strasser[1], Alois Zoitl[2], Franz Auinger[3], and Christoph Sünder[2]

[1] PROFACTOR Research, Im Stadtgut A2, 4407 Steyr-Gleink, Austria
thomas.strasser@profactor.at
[2] Vienna University of Technology, Gußhausstr. 25-27, 1040 Vienna, Austria
{zoitl, suender}@acin.tuwien.ac.at
[3] University of Applied Science Wels, Wels, Roseggerstrasse 12, 4600 Wels, Austria
f.auinger@fh-wels.at

Abstract. Adaptive Manufacturing is identified by the European High-Level Group (Manufuture2004) as one of four major drivers of industrial technologies in "Manufacturing 2020". New technologies for efficient engineering of safe, fault-tolerant and downtimeless systems and their adaptations are preconditions for this vision of future manufacturing. Without such solutions, engineering adaptations of the Industrial Automation and Control Systems (IACS) will by far exceed the costs of engineering the initial system and the reuse of equipment becomes inefficient.

In this work a new approach for model driven, component based development of safe, downtimeless, distributed real–time control and for fault–tolerant, controlled evolution of IACS during their adaptation is proposed. This new method significantly increase engineering efficiency and reuse in component-based IACS.

1 Introduction

Today's markets and economies are becoming increasingly volatile, unpredictable, they are changing radically and even the innovation speed is accelerating. Manufacturing and production technology and systems must keep pace with this trend. Adaptive Manufacturing is identified by the European Manufacture High-Level Group to be one of four major drivers of industrial technologies in "Manufacturing 2020" [1].

Distributed embedded real–time systems for industrial automation and control of plants that evolve towards downtimeless adaptable systems will play a key role to realize the roadmaps towards adaptive manufacturing [2] of products, goods and services in 2020. Most value will then be added in engineering and performing a system evolution rather than in engineering and performing "normal operation".

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 165–175, 2005.

New technologies will be required. Otherwise efforts for engineering the system–adaptations will by far exceed the costs of the initial engineering. Systems that can economically be changed in a controlled evolution and which meet safety constraints and *Quality of Service* (QoS) stability will revolutionize manufacturing of goods, products and services. Downtimes during system evolution need to be avoided

- where costs are unacceptably high—e.g.—especially in systems with concurrent production of many different products with high rate of product version change
- for safety reasons
- where continuous QoS has to be guaranteed to customers.

Controlled evolution of distributed embedded real–time *Industrial Automation and Control Systems* (IACS) is a prerequisite for systems where ramp–up/–down or hold is not a realistic option. For these systems cost efficient engineering of "controlled evolution" of systems and the downtimeless, safe and fault–tolerant execution of the evolution will be crucial.

Component-based reconfiguration of control software is technically hardly possible in current architectures of industrial automation and control systems [3]. This hinders the users from getting the expected technical and economical advantages of reconfigurable systems. The component–based reference–architecture introduced by the IEC 61499 [4] for distributed industrial control systems features first concepts to reach that goal. It defines platform independent reconfiguration services at device level [5,6,7]. The component-based reference-architecture intro-duced by the IEC 61499 [3] for distributed industrial control systems features first basic concepts to reach that goal. It defines platform independent reconfiguration services at device level [4].

The principle challenge and aim of this paper is to present an approach to over–come the limitations of IEC 61499 methodologies and tools. To begin with, chapter 2 discusses the requirements for reconfiguration methods that are necessary for the proposed evolution control engineering. Chapter 3 summarizes the main features and characteristics of IEC 61499 as reference model for distributed automation and control systems with special focus of the management capabilities for reconfiguration. In Chapter 4 an engineering approach for the controlled evolution of control applications will be discussed. Finally conclusions are presented in chapter 5.

2 Requirements for Reconfiguration Methods

This section identifies the basic requirements that have to be met in order to allow controlled evolution, i.e. reconfiguration of control applications, at the control level of automation and control systems. The requirements are introduced from three different viewpoints. One is the domain of control and automation systems. The second deals with the motivations for reconfiguration processes on the device level. The last one introduces the aspect of engineering support necessary for adequate reconfiguration applications.

2.1 Requirements Introduced Through the Domain

In order to provide a reconfiguration support for lower level control systems several requirements have to be met by the programming and modeling languages for control algorithms. Besides technical requirements, some of the requirements derive out of traditions of the control and automation domain. They all can be summarized to [8]:

- *Predictability:* Current PLCs[1] show cyclic execution behavior, which is simple to understand regarding its timing behavior and regarding the flow of execution through the program. Distribution adds complexity here, which has to be hidden from the engineers.
- *Usability:* The control logic has to be easy to understand an maintain as current PLC–based systems. The maintainers of such systems are in most cases plant operators. They usually have only minor or no programming skills.
- *Durability:* System lifespan of IACSs is up to decades, which requires long–term upward and down–ward compatibility in case of replacement of embedded controllers.
- *Form factor requirements:* Small to fit on cheap and robust micro-controllers that can be mounted directly on mechanical components.
- *Standards compliance:* In order to provide interoperability with other components, configurability from any standards–compliant tool, portability at source–level for software–reuse and portability a binary level for migration of software components are required.
- *Real time execution with guarantees:* Interacts with the real world, therefore real-time execution is necessary.
- *Real time reconfiguration with guarantees:* Switching control applications or relocating them during full operation needs planned proceeding regarding reconfiguration and timing for changing from one to another state. The execution environment has to support switching on events and switching on fixed points in time (of "real time" or of a virtual time). Guarantees have to be given that the reconfiguration and switching does not hurt any execution constraints of the running application and that it can be performed within the available and reserved resources (time, memory, power ...)

2.2 Reconfiguration Process Requirements

In addition to the above requirements introduced through the domain, other demands are posed from the appearance of normal and abnormal activities and conditions during execution of application programs. They may motivate reconfiguration processes. As a reaction to special situations of a control system (such as failures in hardware, mechanics, software) the control system needs to be changed and adopted in order to fulfill at least minimal functionality or just degrade to a safe state.

[1] Programmable Logical Controller.

- *Failure detection and system monitoring:* First of all such special situations have to be detected. This may lead to independent supervision or monitoring and diagnostic applications that are executed beside of the normal control functions.
- *Failure diagnosis, selection and initialization of recovery applications:* An algorithm for diagnosis has to find out which failure was detected and feasible steps for failure recovery have to be introduced by starting recovery applications that influence (reconfigure) the main applications in a way so that the main applications can reach safe states or continue operation as intended.

These requirements lead to systems built of independent intelligent modules. The control program in them is composed of migratable components, which are executed independent of each other. But they are equipped with the ability to cooperate for solving control tasks if needed.

2.3 Requirements for Engineering of Reconfiguration Applications

To keep the complex task of engineering reconfiguration applications managable by control engineers the following requirement are introduced through the engineering process.

- *Application centered engineering:* Especially reconfiguration applications have to be considered form the whole applications' point of view. A device centered engineering approach will not be manageable and will lead to complex evolution scenarios.
- *Different engineering views:* Reconfiguration introduces additional views within the engineering. The application of the original system state has to be displayed in comparision to the new system state. And the reconfiguration application and its interconnections to these applications and the sequence of evolution have to be visualized in an intuitive manner.
- *Reconfiguration modelling language:* According to the requirement of Usability in 2.1 also the modelling language for reconfiguration has to be easy to understand and maintain. Therefore a similar semantic has to be used as for "normal" applications.
- *Verification and validation:* Reconfiguration requires advanced methods for verification and validation to a bigger amount than "normal" applications. Any predications have to be stated at engineering time, detection of failures at execution time has to be included into the reconfiguration process. Critical constraints for reconfiguration are processing time and memory consumption according to process the evolution without malfunction of running applications.
- *Version control:* The possibility of describing a evolution step of an IACS enables support of the whole system life cycle within engineering. The continuous system changes may be saved as different versions and enables for instance a undo–functionality.
- *Transition management:* During the process of reconfiguration the actual state of the system is a very important information to be introduced as

transition condition within the evolution sequence. On the other hand the acknowledgement from the previous reconfiguration step has to influence the reconfiguration sequence.

3 IEC 61499 as Reference Model for Reconfigurable Distributed Control

3.1 Main Characteristics of IEC 61499

The programming constructs of today's existing systems, which are mainly based on the standard IEC 61131-3 [9] are too monolithic and closed to fulfil above requirements. The main restriction of current systems is the missing support for building control software out of independent components that can be added and removed at runtime. Therefore new ways for programming and modeling of lower level control systems have to be taken [3].

The new and upcoming standard IEC 61499 "Function Blocks for Industrial Process Measurement and Control Systems" [4] is a reference architecture that has been developed for modeling and implementation of distributed, modular, and flexible control systems. Through the management model [3,10] including the management interface of IEC 61499 compliant devices (as described in chapter 3.2) the new standard provides suitable mechanism for the above mentioned requirements. It specifies an architectural model for distributed applications in industrial–process measurement and control systems (IPMCS) in a very generic way and extends the function block model of its antecessor with additional event handling mechanisms and concepts for distributed systems.

Function blocks are an established concept for industrial applications to define robust and re–usable software components. They can store the software solution for various problems and they have a defined set of input and output parameters, which can be used to connect them to form complete applications. The used network type is not in scope of the specification, but a compliance profile for feasibility demonstrations, which is provided by the Holonic Manufacturing Systems Consortium, specifies its use for Ethernet [11]. The following main features characterize this new standard [12,10]:

- Component oriented basic building blocks called Function Blocks
- Graphical intuitive way of modeling control algorithms done through connecting the in- and outputs of function blocks
- Direct support for distribution
- Definitions for the interaction between devices of different vendors
- Basic support for reconfiguration
- Based on existing standards of the domain

With the constructs and definitions of this standard most of the requirements introduced in the last section can be fulfilled. Because of these features IEC 61499 is suitable as reference architecture for building new agile and adaptive lower level control of next generation automation and control systems.

3.2 Management Interface of IEC 61499 Devices

The configuration of a distributed automation and control system based on IEC 61499 can be enabled by the use of management functions which can be included by each device. For this purpose the standard defines a management application, represented by a device manager function block (depicted in Figure 1). With

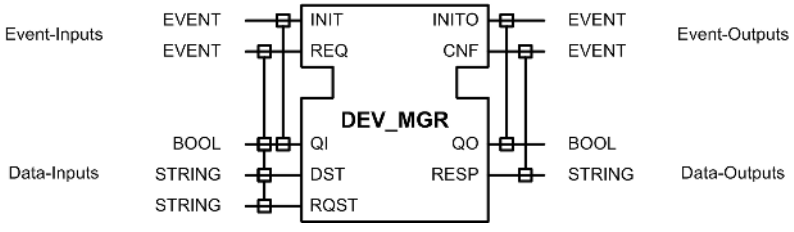


Fig. 1. Device Manager FB

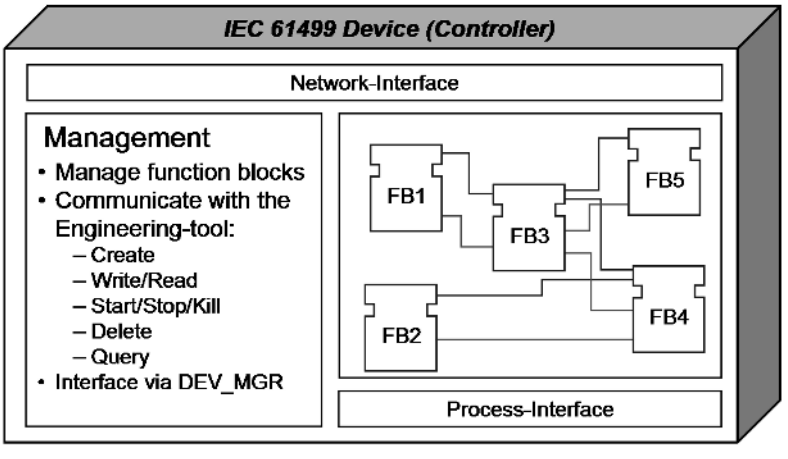


Fig. 2. IEC 61499 compliant device including management functionality and management interface

the use of this block combined with a remote application, which is described in the IEC 61499 Compliance Profile for Feasibility Demonstrations [11], access between different IEC 61499 compliant devices is allowed. This provides a very extensive facility to configure them by sending XML[2] management commands from on to the other. The following standardized functions of the management application can be used to interact with a device:

- Creating function block instances,
- Creating event and data type connections between all types of function blocks,

[2] eXtensible Markup Language.

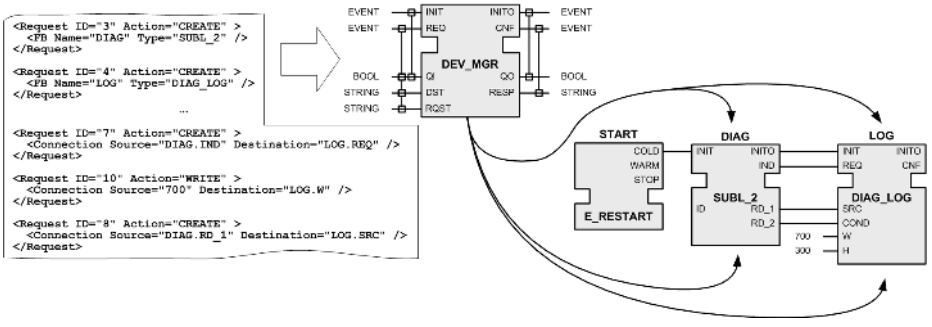


Fig. 3. Configuration example using the Device Manager concept

- Initiating the execution of function blocks,
- Deleting function block instances and connections and
- Additionally it is used to supply status information about the state of the device, the function block instances and their connections

Figure 2 shows an IEC 61499 compliant device with management functionality and the corresponding interface. An example of this service interface is shown in Figure 3 with the corresponding XML based configuration commands. Through the use of XML commands for the communication, IEC 61499 provides a defined interface to other programs. Therefore generic tools can be developed to configure whole IEC 61499 compliant networks, independently from the vendor of the specific devices, although specific parameters have to be provided by the vendors of the devices. This could easily result in a variety of different tools, because no unitary interface is proposed. Probably resulting in increased effort of device reconfiguration, as the handling of different tools has to be learned and the appropriate selected. In addition there are basically the following three classes of device functionality possible which is not standardized in IEC 61499:

1. simple device, pre–programmed
2. simple programmable device, allowing interconnection of a fixed set of function block types
3. user programmable device, which is fully customizable

4 Engineering Methods for Controlled Evolution of Automation and Control Systems

To overcome the limitations of current embedded industrial automation and control engineering methods, we propose an application centred engineering method for efficient component based modeling of applications for controlled, fault–tolerant and safe system evolution of Industrial Automation and Control Systems (IACS). The execution of an evolution will become a normal operational state of such systems, while dependability and QoS of the evolving system are not endangered and migration is cost–efficient. Thus important steps towards Adaptive Manufacturing will be realized.

4.1 Approach for Controlled Evolution

The top–level approach focuses on replacing state–of–the–art "ramp down — stop — download — restart — ramp up" methods with a simple continuous system evolution, which is controlled by an evolution control application that is modeled with components in the same way as control applications are. Evolution control can be either executed from an engineering environment or—if constraints regarding fault tolerance, real–time and safety have to be met — it can be distributed to different controllers (as shown in Figure 4). For safe and fault–tolerant control and evolution execution the reconfiguration application is verified [13] together with hardware capability descriptions of the different devices in the network leading to trustable QoS judgment.

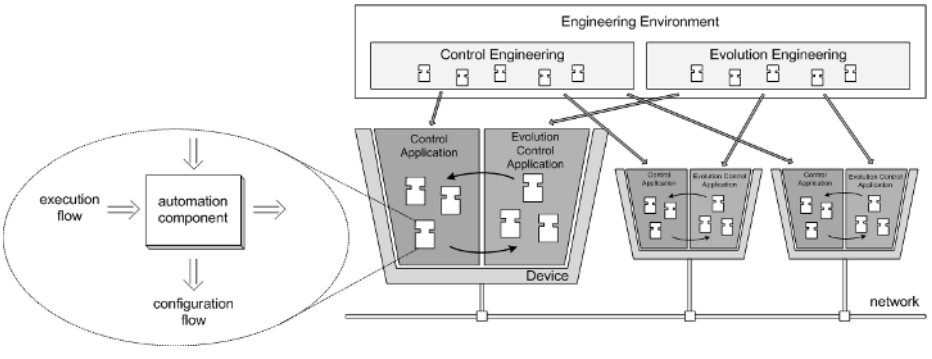


Fig. 4. Evolution control applications for downtimeless system evolution

4.2 Modeling of Control and Evolution Control Applications

The following engineering process provides a method for handling system evolution of Automation and Control Systems by an efficient support for engineering of control applications and evolution control applications. This engineering method consists of the following four major parts as depicted in Figure 5:

1. **Acquire Current System State (Acquire existing application)**
   Collect all data necessary for describing the current system state and deliver it as input to the application modeling. The data consists of the system model including applications currently running in the system, the hardware configuration of the system (used devices and network structure), the mapping of the applications to the different devices and the hardware capability descriptions.

2. **Model New Control Application (Application Modeling)**
   Modeling the new control application based on the existing ones by adding/removing components, their interconnections and specification of application

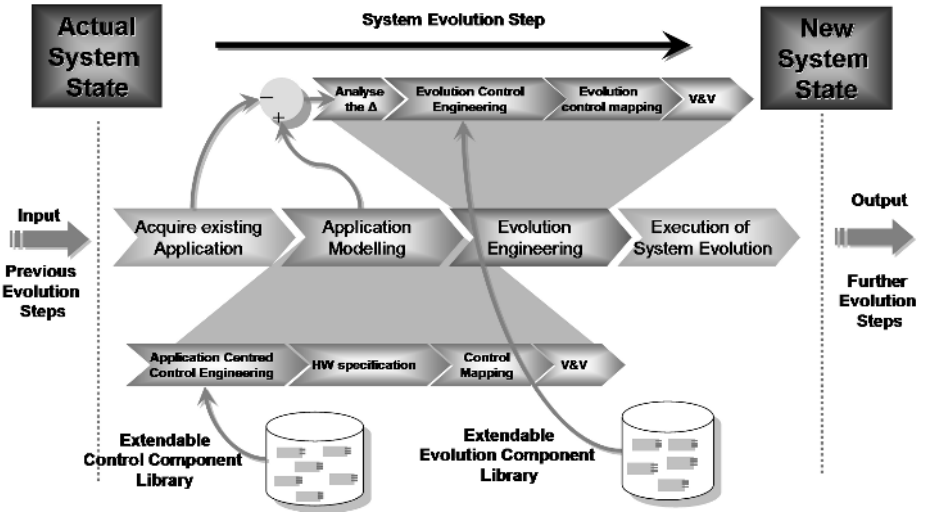


Fig. 5. Engineering process of a system evolution

properties (e.g. real–time constraints). The next step is the configuration of the hardware structure with devices and network connections. After this the modeled application parts are mapped to the according devices they should be executed on. The final step is the verification and validation of the control application in order to determine if the specified constraints will be met.

3. **Model Evolution Control Application (Evolution Engineering)**
With the Delta Analysis differences between the current running control application and the newly modeled control application are determined. These difference serve as input and starting point for the modeling of the evolution or reconfiguration control application that will change the existing application to the new one. The reconfiguration control properties and parameters are specified in the same way as control application properties (according to step 2). Similar to the mapping of the control application the reconfiguration application parts are mapped to the devices. The final step is the verification of the reconfiguration control application together with device capabilities in order to determine if evolution constraints will be met and the running application is disturbed as less as possible.

4. **Execute Evolution Control Application (Execution of System Evolution)**
To execute evolution control applications the utilization of basic reconfiguration services at runtime platforms based on IEC 61499 management commands (cf. chapter 3) is necessary. The first step is to instantiate the evolution control application on the according devices. Next the execution of the evolution control application is done, i.e. the currently running control

application is transformed into the new control application. This is achieved through basic reconfiguration services proving real–time constraint execution of reconfiguration processes at device level. The main services are:

- Control application component load/ remove
- Connect/ disconnect
- Query parameters
- Write parameters
- Query component state
- Write component state

To finish the evolution procedure the evolution control application will be removed after it has successfully executed all commands and all changes are finished.

Based on the basic reconfiguration services of IEC 61499 presented in Chapter 3 efficient engineering methods have to be provided to enable system evolution of IACSs.

5 Summary and Conclusion

The paper introduced a new engineering method for controlled evolution of IACSs. Based on the requirements for reconfiguration methods we described a process for system evolution from an actual system state to new system state. By aggregation of such single steps of system evolution, the whole life cycle of an IACS can be described. The reference model of IEC 61499 and its characteristics especially for management of applications have been demonstrated as basis support for this engineering method.

Acknowledgements

This work is supported by the FIT-IT: Embedded System program, an initiative of the Austrian federal ministry of transport, innovation, and technology (bm:vit). Further information is available at: www.easydac.org

The authors would like to thank the European Commission and the partners of the Innovative Production Machines and Systems (I*PROMS) Network of Excellence for their support under the Sixth Frame-work Programme (Contract No. 500273). PROFAC-TOR is core member of the I*PROMS consortium. Further information about the I*PROMS Network of Excellence is available at: www.iproms.org

References

1. ManuFuture: ManuFuture2003 / ManuFuture 2004 (2003)
2. Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., Van Brussel, H.: Reconfigurable Manufacturing Systems. CIRP **48** (1999) 527–540

3. Martinez Lastra, J., Lobov, A., Godinho, L., Nunes, A.: Function Blocks for Industrial-Process Mesaurement and Control Systems. Number ISBN: 952-15-1244-X in Institute of Production Engineering, Report 67. Tampere University of Technology (2004)
4. International Electrotechnical Commision: Function blocks for industrial-process measurement and control systems. IEC Standard (2005)
5. Brennan, R., Fletcher, M., Norrie, D., eds.: An agent-based approach to reconfiguration of real-time distributed control systems, IEEETransactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures (2002)
6. Fletcher, M., Norrie, D.H.: Realtime Reconfiguration using an IEC 61499 Operating System. In: 15^{th} International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops. (2001)
7. Xu, Y., Brennan, R., et al.: A Reconfigurable Concurrent Function Block Model and its Implementation in Real-Time Java (2002)
8. Zoitl, A., Auinger, F., Vyatkin, V.V., Lastra, J.L.M.: Towards basic real-time reconfiguration services for next generation zero-downtime Automation systems. In: International IMS forum 2004. (2004)
9. International Electrotechnical Commision: Programmable controllers - Part 3: Programming languages. IEC Standard (2003)
10. Lewis, R.W.: Modeling control systems using IEC 61499. Number ISBN: 0 85296 796 9. IEE Publishing (2001)
11. Christensen, J.H.: HOLOBLOC.com - Function Block-Based, Holonic Systems Technology (Access Date March 2005)
12. Christensen, J.H.: Basic Concepts of IEC 61499 (Access Date March 2005)
13. Vyatkin, V., Hanisch, H.M.: Formal-modelling and Verification in the Software Engineering Framework of IEC61499: a way to self-verifying systems. In: in Proceedings of the IEEE Confer-ence on Emerging Technologies in Factory Automation (ETFA'01), Nice. (2001)

Using Radio Frequency Identification in Agent-Based Manufacturing Control Systems

Pavel Vrba[1], Filip Macůrek[1], and Vladimír Mařík[1, 2]

[1] Rockwell Automation Research Center Pekařská 695/10a,
15500 Prague, Czech Republic
{pvrba, fmacurek, vmarik}@ra.rockwell.com
[2] Department of Cybernetics, Czech Technical University in Prague,
Technická 2, 166 27 Prague 6, Czech Republic
{marik}@labe.felk.cvut.cz

Abstract. The radio frequency identification (RFID) is a technology for automatic identification and localization of items, particularly in supply chain. Unlike bar code technology that detects the optical signals reflected from bar code labels, RFID uses radio waves to transmit the information from an RFID tag placed on the physical object to the RFID reader. A vast amount of raw data coming from RFID readers needs to be collected, filtered and preprocessed prior to providing it to high-level applications and information systems. Current architectures (like 'Savant', 'edge servers' and similar) for collection and filtering are most likely of centralized nature. It obviously does not conform to the distributed nature of agent-based solutions for the RFID-enabled manufacturing control systems. In this paper, we present an agent-based solution where specialized agents collect and filter the RFID data obtained directly from RFID readers and provide the data to other agents via standard agent communication mechanisms.

1 Introduction

Although bar codes are the most wide-spread technology for tagging and identification of physical objects used today, this has several restrictions and drawbacks. One of the major issues is the need for line-of-sight between the reader device and the label. It does not often work without human intervention – the United States Postal Service for example estimates the cost of positioning an item for scanning using conventional line-of-sight technologies is about USD 0.04 per item. When multiplying this cost with the number of packages processed within a year (approximately fifty million) and the number of scannings per package needed in a distribution center (about three), they are spending USD 6 million each year unnecessarily [7]. The other problem is that the bar codes are not designed to allow distinguishing between individual instances of objects (all objects of the same type have the same ID).

The *Radio Frequency Identification* (RFID) technology allows remote identification of objects using radio signal, thus without the need for line-of-sight or manual positioning of each item. The passive RFID tags (passive because they do not have

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 176 – 187, 2005.

their own power supply) comprised of a tiny chip and antenna are attached to the physical objects. When the tag enters the range of the RFID reader, it absorbs the energy from the radio field and the microchip, which bears the unique identity code, returns this information back to the reader via modulation of the radio waves (transmission distances range from centimeters to meters). There are also active RFID tags with built-in batteries able to transmit their data over distances up to one hundred meters; however, they suffer from larger size and higher price. The major advantage over the bar codes is that the RFID system allows simultaneous detection of multiple items as they pass through a reader field (for example the presence of all the items in a closed box can be checked without opening it). Additionally, each physical object has its unique ID enabling to easily track and monitor the position of each individual product piece.

However, the RFID tags and readers themselves are only a part of the overall RFID solution. Recently, the so called *ECP Network* architecture [3] has been developed as a global standard for automatic and unique identification of objects in the physical world and their linkage to their virtual representation in networked databases [5]. The EPC Network consists of the following fundamental technology components:

- *The Electronic Product Code* (EPC) – designed to provide a unique identifier for each object. As shown in Figure 1, the EPC typically consists of four segments of binary digits identifying: (a) version of the EPC code, (b) manufacturer company, (c) type of the product and (d) serial number of the product.
- *Low cost tags and readers* – the lowest possible cost can be reached by using disposable read-only tags with a fixed EPC number (usually 64 or 96 bits). Optionally, "empty" tags can be purchased with the ability to write once the arbitrary EPC. In both cases, the information stored in a tag cannot be changed later on.
- *Savant* – the middleware software that is placed between the physical reader infrastructure and higher level information systems (see specification in [6]). Its role is to collect, filter and aggregate the raw EPC data coming from RFID readers and convert them into meaningful form for higher-level applications and information systems. The Savant middleware reduces the volume of information sent to applications – only significant "data events" and summary data packets are propagated to applications and information servers, rather than every individual tag read [5].

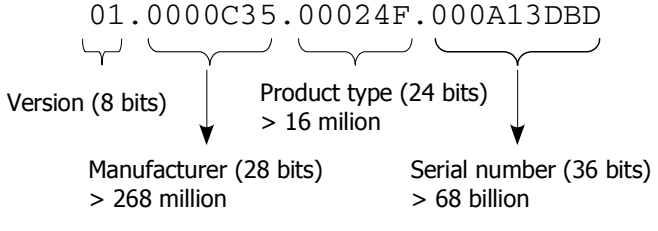


Fig. 1. Electronic Product Code (96 bits version)

- *The Object Naming Service (ONS)* – used to translate an EPC into one or more Internet addresses (URLs) where further information about the object can be found. Typically, these URLs identify an EPC Information Service (see below), although ONS may also be used to associate EPCs with manufacturer's web sites relevant to the objects.
- *The EPC Information Service* – network infrastructure that enables trading partners to share different subsets of their live EPC data through a standard interface, thus without any need to access the underlying databases directly.

In this paper we present our efforts aimed at integration of the RFID technology with the agent-based manufacturing control solutions. The most important aspect of the agent-based control system is, that unlike in classical Computer Integrated Manufacturing (CIM) systems, there is no central control element. The decision-making processes are distributed over a community of autonomous units – *agents* – that are responsible for the local control of particular parts or components of the manufacturing equipment. For instance, one agent represents a CNC machine, the other one controls the movement of the AGV transporting products among the machines and yet another one controls the diverter in the conveyor transportation system. The cooperation between agents to meet the common goals is the other important attribute – the agents interact with each other to simply exchange some information, to request to provide particular services or to participate in complex auction-based negotiations.

In order to integrate the RFID technology with the agent-based control solutions it is reasonable to introduce special mediator agents – the *RFID agents* – that provide the EPC data to the other agents via standard agent communication mechanisms. From this viewpoint, it is not advisable to rely on the centralized plant-wide design of the Savant architecture, where data from all RFID readers are concentrated at single location. The RFID agents could obtain data from such a centralized database and then provide the data to other agents, though such a solution would significantly diminish the major advantages of the agent-oriented approach like robustness and flexibility.

As argued in [2], the use of the centralized Savant is advocated in the literature so far as the most straightforward means of putting an RFID system in-use for manufacturing control. However, other more distributed architectures are also discussed: (i) the *cell-level distribution* where Savant is distributed into different component servers, each serving a separate section or cell of plant operations, e.g. assembly, packaging, material handling etc. and (ii) the *machine-level distribution* where the Savant functionality is distributed at the PLC (Programmable Logical Controller) level enabling one to integrate the RFID information (namely tag sensing) with the real-time control operations.

In our agent-based solution we propose the *agent-level distribution* architecture – the Savant functionality is directly embedded in the RFID agents enabling such an agent to directly collect data from RFID readers, filter the data and store them in its internal real-time storage. The other agents can then subscribe for being informed about particular EPC data events by the RFID agents – the RFID agents have the required data immediately available without any need in obtaining them from any other mediator or centralized database. We suggest that the functionality of the RFID agent of collecting and filtering data from RFID readers is internally ensured by a

special Savant-like module – named *RFID Manager* – plugged into the agent. Since our major programming language for developing agent applications is Java, the RFID Manager is designed as a standalone Java class (with well-defined interface) to be used as a module to other Java applications. Besides the RFID agent, that embeds the RFID Manager, we present also the *RFID Logix Manager* intended as an application running directly on a Rockwell's ControlLogix PLC and providing the RFID data to control applications running on the same controller (similarly to the Savant machine-level distribution architecture).

2 RFID Manager

The core of our RFID-oriented development is a small, self-contained Java application called *RFID Manager*, which task is to collect the RFID data from the readers, provide basic filtering mechanisms and temporarily store the data in its internal memory. The RFID Manager is designed to be used as a module for the development of other RFID-related Java applications. It has a well-defined interface by which it provides the internally stored EPC data to the application that contains/embeds the RFID Manager.

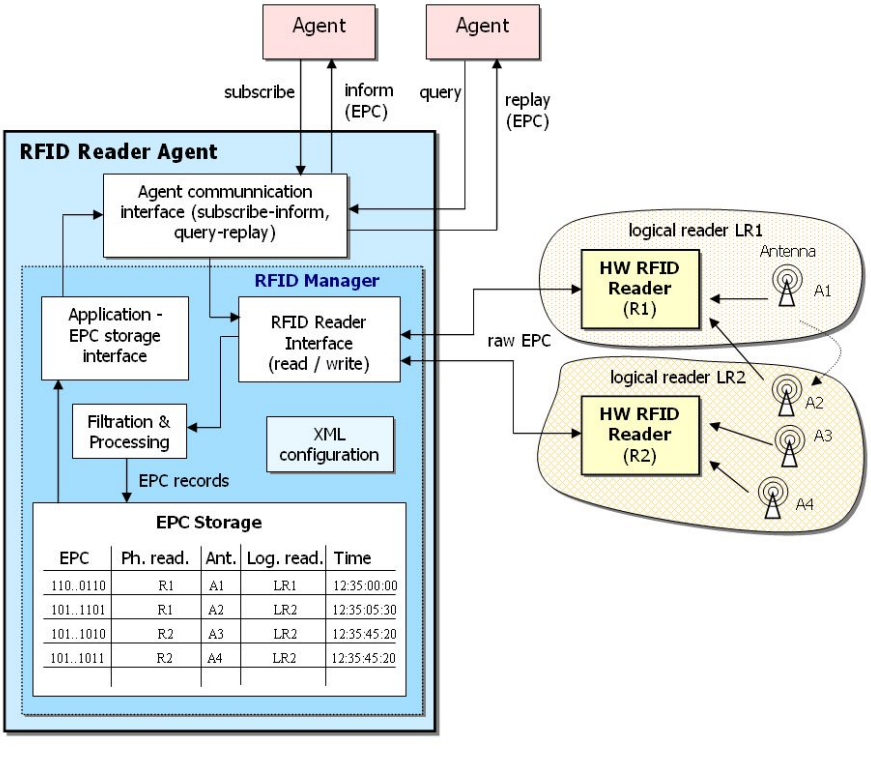


Fig. 2. The RFID Manager application embedded in the RFID agent

As shown in Fig. 2, the RFID Manager consists of four main parts: (i) the RFID readers interface module, (ii) the filtration and processing module, (iii) the EPC data storage and (iv) the application interface.

(i) The first module manages the drivers by which the RFID Manager connects and receives data from physical RFID readers (usually using the Ethernet connection). We have designed a mechanism of pluggable drivers specialized for different hardware readers from different manufacturers (each manufacturer uses a specific communication protocol). An important feature is that these drivers can be dynamically added to the RFID Manager at runtime to enable the connection to newly installed readers. The existing drivers can also be unplugged when a connection to some readers is not needed any more. Additionally, the architecture allows one to implement new reader-specific drivers in the future and to simply add them to the library of drivers without need to modify the rest of the application.

(ii) The filtration and processing component manages EPC data processing modules (filters) that sequentially preprocess/filter raw EPC data retrieved from readers before they are stored in the internal storage. As in the case of the drivers, also filters can be dynamically added or removed at runtime (or just temporarily deactivated) and new application-specific filters can be introduced later on.

The most commonly required type of filtering technique is the *duplicity read* filtering – as the tag enters the RF field of the reader, its EPC is reported by the reader continually (e.g. at 1 sec period) as long as the tag resides in the field. However, it should be interpreted as a single EPC read. Our duplicity filter implementation is based on tagging the EPC reads with timestamps – in case that the time difference between the two reads of the same EPC exceeds given timeout, it is interpreted as a new occurrence of the same EPC in the observed zone (corresponding to a situation when the tag left and then entered the reading zone again). Similarly, the readings of the same EPC that fall into a specified time interval (e.g. 5 seconds) are interpreted as a single occurrence of the EPC (corresponds to a situation when the tag still resides in the reading zone, but it was for example not observed by the reader for a while because of collision or anomaly in the RF field). Some more advanced methods of filtering tag reads over time were described for example in [1].

Another useful processing module that we have implemented is the *Logical Mapping* filter, which allows to define separated reading zones called *logical readers* and associate the information about the physical reader and antenna that actually read

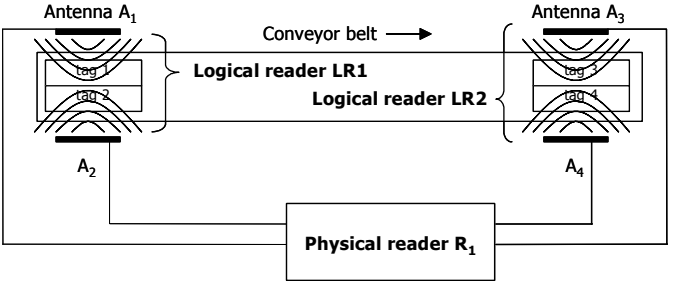


Fig. 3. Example of mapping between physical and logical readers

an EPC with the appropriate logical reader entity. Usually, there can be up to four antennas connected to a single RFID reader and each antenna can be used to read tags at a different location. However, to ensure the high probability of a successful reading, especially in the case of reading large number of tags in one place, it is recommended to use more antennas to cover a particular reading zone.

Fig. 3 shows an example of a conveyor belt with two separated reading zones – the antennas A_1 and A_2 cover the beginning of the conveyor and the antennas A_3 and A_4 its end. Although all four antennas are connected to a single physical reader R_1, this can be viewed as two independent logical readers – LR_1 associated with the antennas A_1 and A_2 and LR_2 associated with A_3 and A_4. Any readings from the antenna A_1 or A_2 are then interpreted by the Logical Mapping filter as readings from the logical reader LR_1, and respectively, readings from antenna A_3 or A_4 are interpreted as readings from the logical reader LR_2.

Fig. 4 depicts another example of the use of the logical readers' mechanism. In this case, eight antennas connected to two physical readers are used to ensure that all tags going through the dock door (e.g. on pallets on a forklift) will be successfully read. Thus, the logical reader LR_1, representing the dock door reading zone, is associated with all the eight antennas – readings from any of the antennas of the physical reader R_1 (i.e. $A_1 – A_4$) or from any antenna of the physical reader R_2 ($A_5 – A_8$) are interpreted as readings from the logical reader LR_1.

Let us stress, that the filters are applied to the EPC data received from readers sequentially. First, the Logical Mapping filter determines the logical reader entity on the basis of the physical reader and antenna attributes of the EPC record. Second, the Duplicity Filter determines, if the same EPC number has already been detected by the same logical reader – if the difference between the timestamps exceeds a given interval, the new EPC record is stored in the storage, otherwise the record is discarded. In the system in Fig. 3 for instance, as the two tags are moving through the reading zone at the beginning of the conveyor, the reader can for example report the following sequence of readings (antenna, tag): (A_1, tag_1), (A_2, tag_1), (A_2, tag_2), (A_1, tag_1), (A_2, tag_2). The fact, that the antenna A_1 was not able to read the tag_2 can be caused, for example, by absorption of the RF waves by the object tagged by tag_1 (usually in the

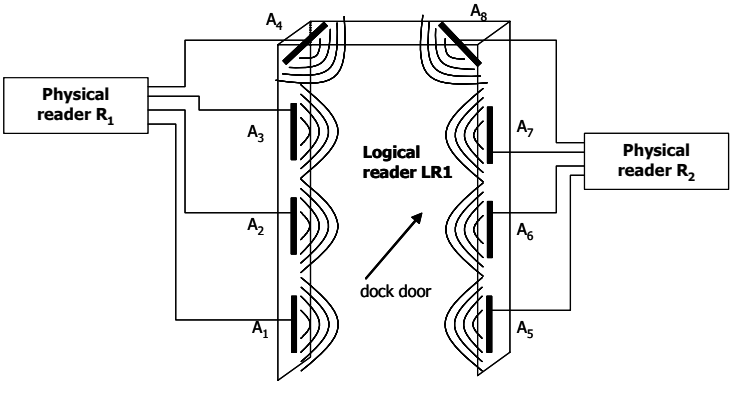


Fig. 4. Single logical reader composed of two physical readers and eight antennas

case when the object contains metal or water). However, such a sequence, when processed by both the logical mapping and the duplicity filters, is interpreted as the single reading of tag_1 by the logical reader LR_1 and the single reading of tag_2 also by the logical reader LR_1.

A significant feature of the RFID Manager is that it allows one to dynamically change the logical mapping definition to reflect the physical changes of the antennas layout. Fig. 2 shows an example, when more antennas were needed to empower the reading capability in the zone represented by the logical reader LR_2. So, the antenna A_2 was physically removed from its current location in the reading zone represented by the logical reader LR_1 and placed to the second zone (however it still remains connected to the physical reader R_1). At same time, such a change was reflected in the logical mapping definition, thus associating the logical reader LR_2 with the antenna A_2 connected to the physical reader R_1 and both the antennas A_3 and A_4 connected to the reader R_2.

(iii) The third module is the EPC storage where the processed data are temporarily stored. As can be seen in Figs. 2 and 5, each record contains not only the EPC number, but also additional information like the physical reader and antenna identification, logical reader recognition, timestamps, etc.

(iv) The *application interface* then provides such data to the application that contains the RFID manager and supported methods include for example "report any new EPC record" (using a subscribe-inform like mechanism), "get all data that correspond to a particular reading zone or to a particular EPC number".

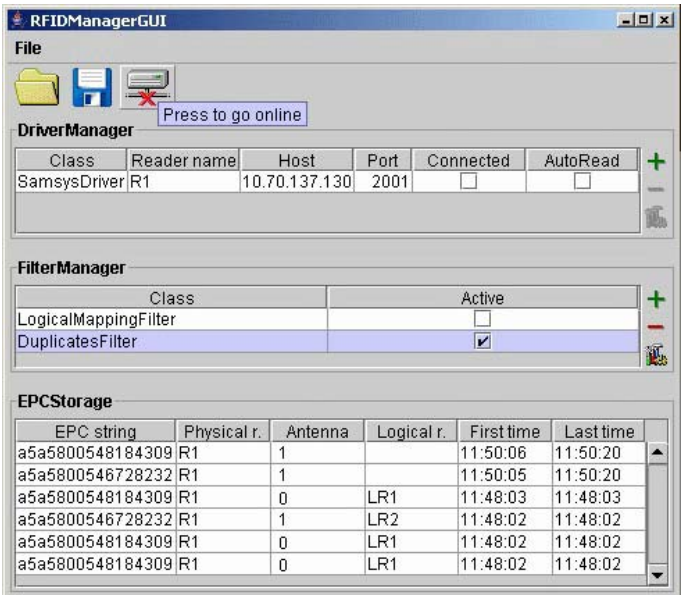


Fig. 5. RFID Manager GUI window enabling smooth interaction with the RFID Manager Application to star, stop, configure, or simply observe the system status in real time

The Graphical User Interface (Fig. 5) is used to observe the status of the RFID Manager. It displays the list of reader driver plug-ins, list of all installed filters and the table of the last received and processed EPCs. The RFID Manager can also be controlled and fully configured via the GUI – the readers can be switched on and off, the filters can be activated or deactivated, and new drivers and filters can be added at runtime etc. The current configuration of the RFID Manager can be saved to and loaded from an XML file.

Both the GUI and the RFID Manager are designed to run separately – it is supposed that the RFID Manager will run on a simple hardware with Java support, for example on an industrial PLC or directly on the RFID reader hardware. The GUI can run on a smart display or personal computer and communicate remotely with the RFID Manager via the Ethernet network.

Currently, the RFID Manager is used in two different applications:

- The *RFID agent* that provides the EPC data to other agents via standard agent communication mechanisms (see next Section)
- The *RFID Logix Manager* that provides the EPC data to the control applications running on an industrial PLC (Rockwell Automation ControlLogix™) by writing them directly into the PLC data-table (common data memory).

3 Integration of the RFID with Agent-Based Control

The pilot Rockwell Automation application where the RFID technology is integrated with the agent-based approach is the agent-based solution for the material-handling task. Aimed particularly at the transportation of discrete workpieces (products) among the manufacturing cells (machines) on the factory shop floor using a conveyor-based transportation, we have developed the agent-based simulation and control system called MAST – Manufacturing Agent Simulation Tool.

In this system, we have designed and implemented the following basic set of agents for elementary material handling components (for more details see [9]):

- The *work cell* agent that represents a general manufacturing cell, e.g. a drilling/milling machine, storage area, assembly machine, docking station, etc. Work cells play roles of *source* and *destination* components between them the workpieces are transported in the material handling system.
- The *conveyor belt* used to transport workpieces between two other components which it is connected to.
- The *diverter (crossing)* agent that switches workpieces between the conveyor belts in order to navigate them to desired destination work cell. Each diverter agent holds an up-to-date routing table containing the knowledge about the destination work cells reachable via each of its output conveyors (the routing tables are determined by cooperation of the agents – see more details in [9]).

The need to integrate the RFID technology in the MAST environment arose as we started to think about the application of the MAST tool for the simulation and control of the Holonic Packing Cell [4]. The Holonic Packing Cell has been built at the Institute for Manufacturing at the Cambridge University to provide a physical manufacturing

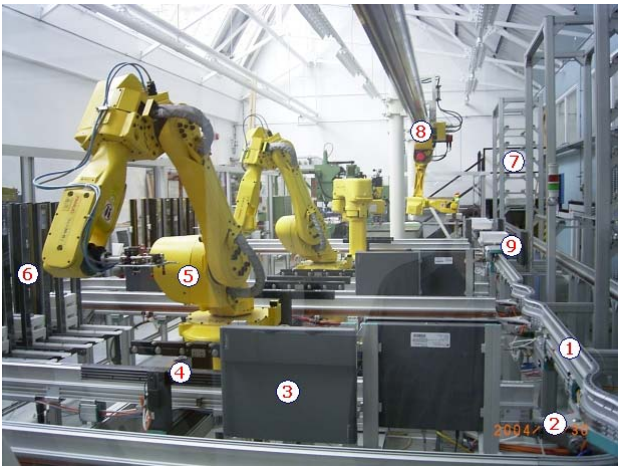


Fig. 6. The Cambridge packing cell: (1) Montech conveyor loop, (2) gate, (3) RFID reader, (4) docking station, (5) Fanuc M6i robot, (6) storage area, (7) rack storage, (8) gantry robot, (9) shuttle

environment for experiments with the RFID technology and agent-based control. As shown in Fig. 6, the lab physically consists of (i) the conveyor loops (Montech track) used to transport shuttles carrying workpieces (Gillette gift boxes); there is one main feeding loop (labeled by number 1) and two subsidiary loops (orthogonal to the main loop) leading to robots where the boxes are packed, (ii) two gates (one of them labeled by 2) that navigate the shuttles out of the main loop to the subsidiary loop and vice versa and (iii) the RFID readers (labeled by 3) placed in front of the gates to read the EPCs of incoming shuttles to properly navigate them through the gate.

The Cambridge cell-related extensions of the MAST tool were particularly related to the modification of the diverter agent so that it is able to represent the gate, i.e. to control the switching of workpieces (shuttles) on the basis of the information from the RFID readers. Indeed, it was also necessary to implement the RFID agent that polls the RFID reader and passes the information about read shuttles' EPCs to the gate agent.

Fig. 7 gives a schematic overview of a general layout of components that take part in the RFID-based workpiece switching in a conveyor transportation system. Generally, workpieces come to the diverter (D_1) from two different "input" directions (in this case from the work cells W_1 and W_2) and can be routed to two different "output" directions (to reach the W_3 or W_4 destination work cell). The detection system by which the diverter agent is informed about incoming/leaving workpieces is based on the notion of *symbolic locations* [8] – there are two particular locations (reading zones) in front of the diverter (SL_1 and SL_2) and two other zones (S_3 and S_4) behind it, where the presence and, in the former case, also the identity of workpieces have to be detected. As a matter of fact, the terms "symbolic location" and "logical reader" (described in previous Section) match each other.

It is suggested that for each symbolic location in the system there is a particular agent that provides the detection capabilities to other agents. Such an agent registers

in the Directory Facilitator (a common component of the multi-agent system providing yellow-pages services) its detection service together with the information on what symbolic location(s) it is in charge of. Any other agent can then query the Directory Facilitator to get the name of the agent that is responsible for a particular location and then subscribe this agent for being informed whenever any workpiece is detected at that location.

Fig. 7 shows the current agent implementation in the MAST system, where the detection of workpieces in front of the diverter is done by a real RFID reader (product of the Samsys company) with two antennas while the rest of the system (conveyors, work cells, diverter and sensors) is simulated. The RFID reader reports the EPC data via the Ethernet to the RFID Manager that is embedded in the RFID agents. The *Logical Mapping filter* in the RFID Manager obviously defines two logical readers, one associated with the first antenna used to cover the SL_1 zone and the other one associated with the second antenna used for the SL_2 zone (in case we would use four antennas, two of them could be associated with the first and the two others with the second logical reader).

The two RFID agents register in the Directory Facilitator their detecting capabilities associated with the SL_1 and SL_2 locations respectively. Thus the diverter agent (D_1) easily discovers the names of the RFID agents that are in charge of these locations and subsequently subscribes to them for being informed about the EPC codes of incoming workpieces. The same architecture also applies for the "output" locations S_3 and S_4 where the presence of workpieces is detected by simple sensor agents also registered in the Directory Facilitator and subscribed by the diverter agent.

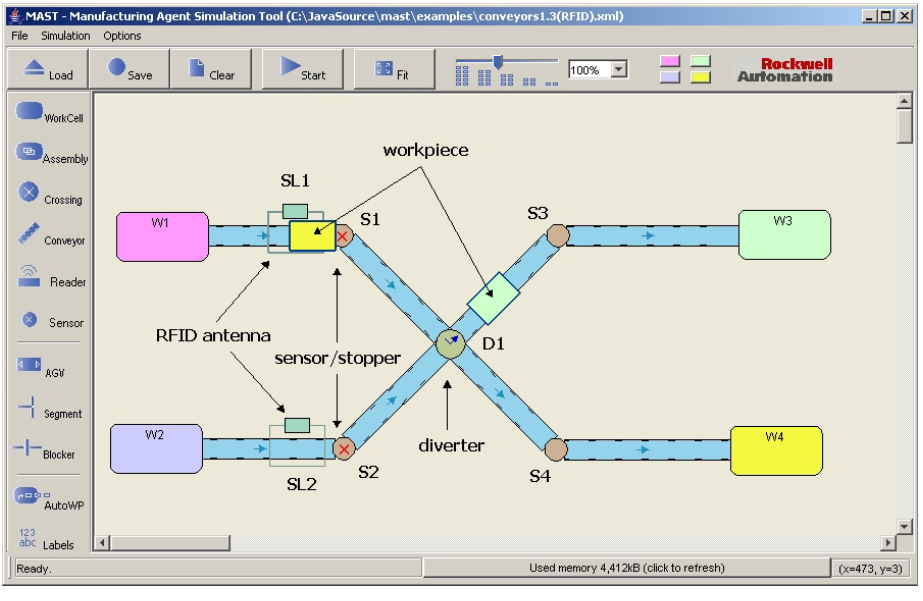


Fig. 7. Layout of components in the RFID-based navigation in a conveyor transportation system

So, when a workpiece enters the SL_1 or SL_2 zone, the associated RFID agent sends a message to the diverter agent about the EPC of the workpiece along with the indication of the reading zone. The issue is, how the diverter agent finds out what is the desired destination of the workpiece to be able to switch it properly – such information cannot be stored directly in the EPC number since the tags are usually read-only. One possible solution would be to introduce a special agent that is aware of the destinations of all currently transported workpieces. Nevertheless, such a centralized component is not desirable in the multi-agent system (where we do not like to see any central element).

The solution that we suggest is based on the consideration that each individual physical workpiece is represented by an agent. Such an agent is usually designed as a smart entity that proactively fulfills its own objectives. Basically, it negotiates with the machine agents to have a specific set of operations performed upon the workpiece/product as well as ensures the transportation of the workpiece between these machines. Thus, while the workpiece is being transported, the associated workpiece agent knows exactly what is its destination, i.e. the name of the next work cell where the workpiece will be processed.

Moreover, it is suggested that the name, at which the workpiece agent can be contacted in the multi-agent system, is same as the EPC number of the physical product. Then, when the product approaches the diverter and the RFID agent informs the diverter agent about its EPC, the diverter agent sends a request directly to the workpiece agent (using the EPC as its name) to retrieve the product's destination. Let us stress, that in the fully agent-based manufacturing control architecture, all the agents are expected to run directly on PLCs. In such a case, it is highly recommended to run also the workpiece agents on the PLCs (instead of placing them on a single/centralized computer). The mobility feature of the agent runtime environment enables, that the workpiece agent can move itself (including its program code and state) among the PLCs to be physically as close as possible to the product as well as to the machine agent(s) that control the actual manufacturing operations.

4 Conclusion

The paper presents a pioneering approach to using the RFID technology in manufacturing control applications by integrating it with the multi-agent system approach. Unlike in traditional centralized Savant-like architectures, where the RFID data from all readers are concentrated in a centralized database, we present the architecture of distributed *RFID agents* that retrieve the RFID data directly from readers, filter them and store them in their local storages. Each RFID agent is supposed to be responsible for collecting the EPC data from a specific reading zone(s)/location(s). Such a service is registered in the Directory Facilitator so that the other agents that are interested in the RFID data from a particular location can easily find out the corresponding RFID agent.

One of the main features of such a solution is a high degree of flexibility and dynamic reconfigurability. Firstly, new RFID readers can be easily integrated to the system at runtime as there are, for instance, new transportation paths or new machines added to the shop floor. In such a case, a new RFID agent can be created and associ-

ated with new reading zones or the existing RFID agent(s) can take the responsibility for these zones (extending its/their information registered in the Directory Facilitator).

Secondly, the functionality of RFID readers can dynamically change – the proposed architecture allows the user to physically move a particular antenna from one reading zone to another one and to reflect such a change in the RFID agents. For instance, when an RFID agent detects a failure of a particular antenna that is used alone to cover a specific reading zone, it can inform the subscribed agents about the temporary inaccessibility of the EPC data from this zone. In this case, the reconfigurability of the Logical Mapping filter (part of the RFID Manager embedded in the agent) allows the user to remove physically a redundant antenna from another zone and move it to the "failed" one – the subscribed agents are then informed by the RFID agent that the reading zone is up again.

References

1. Brusey, J., Harrison, M., Floerkemeier, Ch., Fletcher, M.: Reasoning about uncertainty in location identification with RFID. In: Proceedings of Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico (2003)
2. Chokshi, N., Thorne, A., McFarlane, D.: Routes for Integrating Auto-ID Systems into Manufacturing Control Middleware Environments. White Paper CAM-AUTOID-WH026, http://www.autoidlabs.org/researcharchive, Auto-ID Center, Institute for Manufacturing, University of Cambridge, UK (2003)
3. EPC Global homepage, http://www.epcglobalinc.org
4. Fletcher, M., McFarlane, D., Lucas, A., Brusey, J., Jarvis, J.: The Cambridge Packing Cell - A Holonic Enterprise Demonstrator. In: Mařík, V., Müller, J., Pěchouček, M. (eds.): Multi-Agent Systems and Applications III, LNAI 2691, Springer Verlag, Berlin, Heidelberg (2003) 533-543
5. Harrison, M., McFarlane, D., Parlikad, A. K., Wong, C. Y.: Information Management in the Product Lifecycle-The Role of Networked RFID. In: Proceedings of the 2nd IEEE International Conference on Industrial Informatics INDIN 2004, Berlin, Germany (2004) 507-512
6. OatSystems& MITAuto-IDCenter. The Savant. Technical Manual MIT-AUTOID-TM-003, http://www.autoidlabs.org/researcharchive, MIT Auto-ID Center, Cambridge, MA, USA (2002)
7. Philips Semiconductors, RFID Delivers the Good, On the Move Magazine, Vol. 4, Issue 2, http://www.semiconductors.philips.com/news/publications/index.html, (2002) 12-13
8. Römer, K., Schoch, T., Mattern, F., Dübendorfer, T.: Smart Identification Frameworks for Ubiquitous Computing Applications. In: Proceedings of IEEE International Conference on Pervasive Computing and Communications, Texas, USA (2003) 253-262
9. Vrba, P. MAST: Manufacturing Agent Simulation Tool. In: Proceedings of IEEE Conference on Emerging Technologies and Factory Automation, Vol. 1, Lisbon, Portugal (2003) 282-287

Resolving Scheduling Issues of the London Underground Using a Multi-agent System

Rajveer Basra[1], Kevin Lü[1], George Rzevski[2], and Petr Skobelev[2]

[1] Brunel Business School, Brunel University,
Uxbridge, Middlesex, UK, UB8 3PH
[2] Magenta Technology, 33 Glasshouse Street,
London, UK W1B 5DG

Abstract. We consider the scheduling issues for The London Underground (LU), which spans the entire city, and is a vast network of inter-related railway lines. By its very nature, it is a dynamic, complex and unpredictable environment; operating or being maintained 24 hours-a-day. This paper reports on an investigation into how a Multi-Agent System (MAS) may be used for resolving scheduling issues for LU. It is a previously unexplored domain. A prototype system MASLU is developed through the use of MAS technology, in an innovative and unique manner, with a view to resolving the London Undergrounds scheduling issues in real time.

Keywords: Artificial Intelligence (AI), Multi-Agent Systems (MAS), Intelligent Agents (IA), London Underground (LU), Scheduling.

1 Introduction

The London Underground (LU), popularly known as The Tube, spans nearly 400 route miles, with the busiest line of the system carrying over 180 million passengers per year [1]. This traffic places a considerable pressure upon LU logistics systems, which must ensure that all trains are running on time, with crews in place and all required resources properly allocated. In addition, logistics must cope with the occurrence of unpredictable events such as track or train failures, no-show of crews, accidents and many others. The LU is a highly dynamic, complex, and uncertain environment, and as such represents a formidable logistics problem. Academic researchers and engineers have investigated alternative approaches for train scheduling as reviewed in [4]. The planning process (especially for busy passenger trains) is segregated over several stages [5]. Initially train operators such as Metronet or Tubelines create outline draft timetables for the services [6]. These estimates are based upon approximation of resource requirements. These timetables are then adjusted and revised to eliminate all conflicts. Nevertheless this is generally a heuristic approach and this slow ad hoc process does not allow the operators to investigate 'what-if' options due to time constraints. As addressed by [7] there are several advantages of computer based algorithms over manual methods, the principle of which is that results can be prepared in a fraction of the time that is required by manual schedulers, thus allowing for a greater number of options to be examined [4]. It is these systems however, that have provided the motivation for this research as

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 188 – 196, 2005.

they represent what has been achieved through the application of an automated approach, yet an automated approach has limitations especially when applied to such a complex, dynamic and unpredictable environment such as that of the LU, thus the system implemented in this paper (MASLU) illustrates the results and benefits that can be reaped through the innovative application of an intelligent system in this previously unexplored domain through.

2 London Underground and the Existing System

The London Underground (LU) is a complex and dynamic infrastructure with very demanding logistic and scheduling requirements, which need to be met daily, on a real-time basis [1]:

- LU operates over nearly 400 route miles of underground railway line, the majority of which is double track.
- The busiest line on LU is the District Line carrying over 180 million passengers per year over its 40-mile length.
- The busiest station on the LU network is Victoria, closely followed by Oxford Circus, both of which are the start or end points for over 85 million passengers each year.
- During the three-hour morning peak, 34,000 people enter Victoria,
- Every LU train travels the distance of 75,000 miles a year.
- The LU carries 19 million individual passengers making over a billion trips a year.
- 150,000 people an hour enter the LU system.
- Any logistics system would have to cope with staff illnesses and no-shows as well as with frequent track and train failures.
- LU stops train services during the night for maintenance. Could a sophisticated real-time maintenance logistics system enable LU to operate 24 hours a day 7 days a week?

The logistical problems of the LU system provide an exceptional example of an environment that is uncertain, unpredictable, complex and dynamic. It is far more advantageous that an intelligent system should be developed instead of employing human experts to solve these hindrances.

A database has deployed for timetabling and operation plan generation. In addition, more functions have been introduced based Workflow Management Application technology, it can provide based automated Task Management, Resource Management and Allocation Function for the Scheduling Department [5]. The above conventional methods (including the CART/WMA applications) allocate resources to demands by following pre-determined algorithms, thus operating in a strictly defined sequential manner. When dealing with large quantities of resources and demands that frequently change, the time required to accomplish the allocation is rather excessive because after every change the allocation process begins from the beginning. When the frequency of changes is high, the optimisers tend to oscillate, consequently never obtaining an optimal solution. Centralised Intelligent Systems are somewhat better because they are powered by heuristics, and therefore faster, but still inadequate when

dealing with frequent or repeated changes [6]._Although CMC has managed to provide LU with an automated system the system is still based around an Oracle database that requires the expertise of manual schedulers. Despite reducing the burden placed upon the operators the capabilities for exploring "What-if" options is still limited. Current approaches to scheduling have enabled LU to a level of automation [2] [3], however this approach differs from an Intelligent approach such as that adopted by a MAS.

3 MAS: A Promising Solution

A multi-agent system consists of a collection of intelligent software agents. They are intelligent in the sense that they can make decisions under conditions of uncertainty. These agents are capable of accomplishing their tasks through interaction with other artificial intelligence agents or humans. An agent can be defined as a particular kind of physical or logical entity that presents the properties of autonomy, reactivity and proactiveness. A MAS must therefore have autonomous agents functioning in parallel and attempting to achieve a goal or to fulfil a satisfaction function, furthermore these intelligent agents must possess a high-level interaction mechanism independent of the problem to be solved, such as communication protocols and mechanisms that enable the agent to interact with its environment. Consequently in a MAS, a software agent essentially does what one expects of a reasonable, advanced program: it is embedded in an environment in which it is capable of achieving certain tasks with some degree of autonomy, i.e. without constant human guidance or intervention. MAS solve specified tasks through negotiation among agents and therefore it is the effectiveness of this interaction that determines the effectiveness of the system, rather than sole performance of each agent. Systems that saturated with a large quantity of agents that are able to interact with each other and users, provide a powerful, cost-effective problem-solving tool.

4 A MAS Based System for London Underground (MASLU)

A prototype system (MASLU) for resolving scheduling issues on the London Underground has been developed based on multi-agent technology [9]. MASLU is aimed at offering the users an effective means of allocating resources. It allows the user the opportunity to first input data, highlighting requirements, such as the stations included and the time-allotment for each stop. Based upon this knowledge, MASLU agents, acting in a virtual world are able to make decisions using their ontology, then guide users upon the most suitable allocation of resources and generate schedules based upon this information.

4.1 Problem Analysis

The work began with the problem analysis aiming to identify key concepts that define the LU domain knowledge. This involved role modelling, a role describes a set of entities that can occupy the same position in a reoccurring structure, the attributes of each entity must then be accurately identified and assigned enabling the agent to

accurately fulfil its key role within the system. Thus it is the purpose of this stage that invokes the developer to deliberate the needs of each agent, consideration was given to objects such as train, track, crew and station, and to the relationships between these objects.

4.1.1 Ontology of MASLU

Based on detail requirements specification and the role models, the MASLU ontology was created. The first task was to define classes of objects and relationships between objects that are of interest. Key classes of objects and relationships are illustrated below:

ROUTE: A railway line on which a train moves. The initial and terminal stations define the route.
SECTION: A segment of the track that connects two stations.
LOCATION: A station at which the train must stop. Locations and sections compose a route.
DRIVER: A resource that must be assigned to a route.
CARRIAGE: A resource that must be assigned to a route.

MAS match resources to demands. These matches can be either full or partial. As new demands and resources are made available agents perform re-matches if required. The matches may be the result of competition or collaboration amongst the agents. A number of demand- supply relationships have been created;

SECTION	←→	ROUTE
ROUTE	←→	LOCATION
LOCATION	←→	SECTION
CARRIAGES	←→	DRIVER
DRIVER	←→	ROUTE

4.1.2 Agent Architecture

Once all objects have been identified, consideration must then be provided to the structure of the agents and how they shall communicate to ensure that the Agents are capable of fulfilling the requirements placed upon them by the user and the system. The design principles are introduced as following, which ensure that they are capable of fulfilling their system objectives and provide the programmer with a blueprint. Within the system there will be three key types of agents, however since the:

- DF (DIRECTORY FACILITATOR) AGENT: This agent is the agent that provide the yellow pages service to all other agents, thus this agent contains information on all other agents within the system, providing a means by which an agent can find all other agents that provide a service that the agent may require. This agent is necessary within the system if it is to comply with FIPA (Foundation for Intelligent Physical Agents) specifications [10]. This agent furthermore enhance the speed of the system as it will ensure that only the

required agents start renegotiations in the event of new resources becoming available, and not the entire system.

- DEMAND AGENTS: These are agents that will attempt to acquire resources for their users; this can take the form of an agent representing a train that may require a resource such as a section of track (route), a driver or any other relevant resource. The demand agents will find both a primary match (Full matching) and reservation match (Partial matching), to ensure that the next best alternative ids declared at all times
- SUPPLY AGENTS: The supply agents are the agents that will represent the resource to ensure that this resource is provided to the most suitable demand. The Supply agent however can refuse to assign the resource to a buyer agent, this ability to refuse, is explicit to MAS, and indicates a key difference between MAS and Object-orientated systems.

Since the DF Agent is only necessary to maintain the integrity of the system and manage its functions the Demand and Supply agents will be doing the core of the negotiations, and there structure is as follows:

DEMAND AGENT

A new resource becomes available / A current resource becomes unavailable,

Initialise Agent and set requirement criteria,
DO Search in DF Agent for all matching flags,
IF Any matching flags,
THEN set no of matches = n,
DO Set Search = 0,
Search resource criteria,
IF Resource meets any requirements,
THEN Set as Full Match
IF Full Match already set,
THEN Compare Full Match with new resource,
IF New resource meets greater number of requirements,
THEN Set as Full Match,
ELSE Set as Partial Match,
IF Partial Match already set,
THEN Compare Partial Match with new resource,
IF New resource meets more requirements,
THEN Set as Partial Match,
ELSE Reject Resource,
Search +1,
WHILE Search < n,
ELSE Terminate agent,
WHILE No new resource becomes available / A current resource becomes unavailable.

SUPPLY AGENT

A new resource becomes available,

Initialise Agent and set requirement criteria,
Register Agent with DF Agent and set Flag,
Search DF agent for number of Demand agents,
DO Set number of Demand Agents = n
Set Agents Made Contacted = 0
Wait for Demand agent to contact,
Compare requirements,
IF Demand Agent meets requirements,
THEN Assign resource to Demand Agent,
ELSE Reject Offer,
Agents Made Contact + 1,
WHILE Agents Made Contact < n,
Terminate Agent.

4.1.3 Scenes

In MASLU, *Scenes* are implemented in the virtual-world in order to represent real-world situations. They specify the agents and actions that agents are to take. *Scenes* can be created in advance and saved or implemented at the runtime. The possible number of scenes that can be generated is infinite since the number of possible situations that can be encountered in the real world is endless.

4.2 Constructing the Ontology

Prior to the virtual-world and scenes being created and implemented in MASLU, the ontology must be constructed, since this shall define the parameters and policies that the agents and virtual-world must adhere to. Initially *Objects* must be created, as in any object-orientated approach, the *objects* form the means by which the data can be displayed and manipulated. Each *object* is then assigned a number of *Attributes*; this meta-data will be used to add additional information and code elements hence expanding the functionality of the *objects*. Once all *objects* have been assigned the appropriate attributes required to define them in the virtual-world, the *Interface Behaviours* must be set. The *interface behaviours* are implemented to connect the *object* to the *scene,* thus enabling it to be displayed correctly and importing all assigned *attributes.* The process thus far has created the *Descriptive Ontology*, however this alone in insufficient for the system to operate since no agents have been created to implement the ontology, nor have any demand or resource relationships been set. Consequently the *ontology of the world's demands and resources* must be constructed. This involves assigning demand and/or resource agents to each of the *objects*. Once Demand and Resource agents have been created *Decision-Making Conditions* must be composed and *Event-Processors* defined in the *Decision-Making Machine (DMM)*. Hence the DMM contains the conditions and scripts that dictate the possible behaviours and communication of the agents in addition to creating templates for matching relationships. The DMM is the instrument through which all information of the descriptive ontology is passed to the agents that are then implemented in *scenes* in the *virtual-world* in order to mimic real-world situations.

4.3 Dynamic Rescheduling

In this section, an example (*Dynamic Changes to Schedule)* of resolving logistics issues on LU is presented to illustrate capabilities of the MASLU.

The objective of this scene is to demonstrate the ability of the system to react to a late addition of a high-priority service by dynamically altering schedules of trains currently travelling on the same line. Each service has a driver and carriages and the stations Osterley, Barons Court and South Kensington have been specified as suitable for the overtaking. Three services are created with a five-minute gap between them (Fig. 1), their schedule can be seen in Fig. 2 and Fig. 3. The new dynamically created schedule showing how the newly added express overtakes slow trains at appropriated stations Fig.4(a).

The load levels for each station are illustrated in (Fig. 4(b)). This informs the user of the usage level of each station and section on the route.

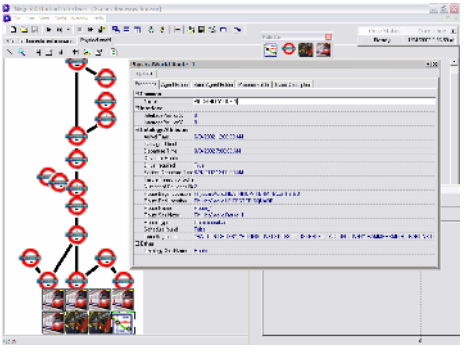


Fig. 1. Route attributes

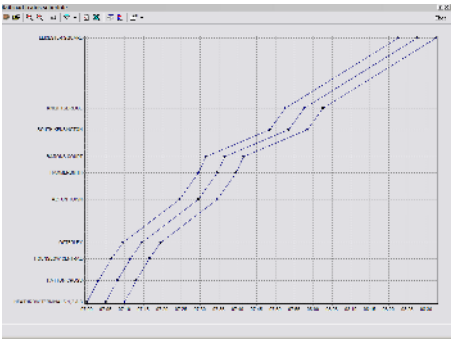


Fig. 2. A diagrammaticview of the schedule

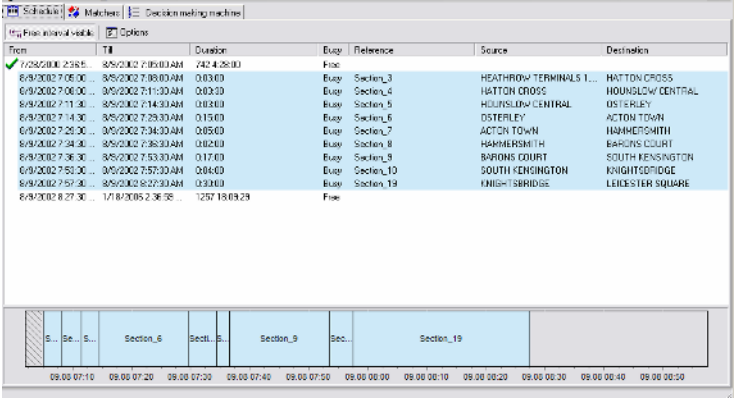
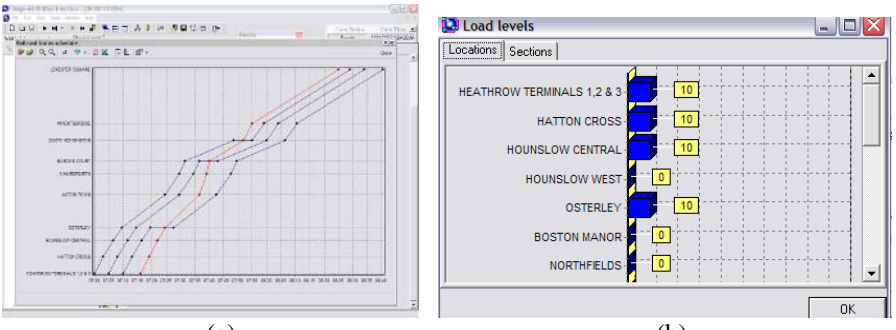


Fig. 3. An alternative view of the schedule



(a) (b)

Fig. 4. The loading of stations and sections of the route

5 Conclusion

MAS are a promising technology to tackle scheduling problems characterised by frequent changes and uncertainty. The paper demonstrates some of the features of a powerful multi-agent system when applied to scheduling problems of The Tube, which represents a highly complex, dynamic and unpredictable environment. Three typical scheduling situations have been modelled and simulated demonstrating how MASLU can resolve serious logistics issues in real-time with no intervention required from human operators, thus reducing the burden placed upon Logistic staff.

The MASLU described in this paper is but a small example of what could be done with this powerful technology. Further work could include modelling the whole LU transportation system and producing in real time dynamic schedules for LU trains, crews and maintenance. For such an increase in scope it would perhaps be necessary to spread the computational load over a number of workstations in a parallel manner.

See that the communication costs and workload skews of four cases are quite close. Because the weight used in the XGP algorithm is dependent on the query frequency, the partitions for different query distributions will change accordingly.

References

[1] Official Website For The London Underground Infrastructure. *www.TheTube.com*

[2] Ernst A. T., Jiang H., Krishnamoorthy M. and Sire D. (2004) Staff Scheduling And Rostering: A Review Of Applications, Methods And Models. European Journal of Operational Research 153 (2004), 3-27.

[3] Cmc limited; Case Studies: *http://www.cmcltd.com/case_studies/transportation/lul.htm*

[4] Matta R. and Miller T. (2003) Tproduction And Inter-Facility Transportation Scheduling For A Process Industry. European Journal of Operational Research 2003.

[5] Watson, R. (2001) The Effect Of Privatisation On Train Planning: A Case Study Of The UK. Transport Reviews 21 (2), 181-193.

[6] Sun Micro-Systems: Customer Success Stories. *http://www.sun.com/smi/Success/Industry Specific/Transportation/London.Under.html*

[7] Carey, M. & Carville, S. (2003) Scheduling And Platforming Trains At Busy Complex Stations: Transportation Research: Elsevier Science Ltd
[8] Hernandez J., Ossowski S, and Garcia-Serrano A. (2002) Multiagent Architectures For Intelligent Taffic Management Systems. Transportation Review Part C, 10 (2002) 473-506.
[9] *http://www.magenta-technology.com*
[10] Wooldridge, M. (2002) An Introduction To Multi-Agent Systems: John Wiley & Sons ltd: Chichester.

KARMEN: Multi-agent Monitoring and Notification for Complex Processes

Larry Bunch, Maggie Breedy, Jeffrey M. Bradshaw, Marco Carvalho, and Niranjan Suri

Florida Institute for Human and Machine Cognition,
40 S. Alcaniz St. Pensacola, FL 32563
{lbunch, mbreedy, jbradshaw, mcarvalho, nsuri}@ihmc.us
http://www.ihmc.us/

Abstract. Early and consistent detection of abnormal conditions is important to the safe and efficient operation of complex industrial processes. Our research focuses on enabling the operators and engineers who control and maintain such systems to describe process conditions to software agents, deploy such agents to continuously monitor live process data, and receive appropriate notification from their personal agents concerning the process state. The resulting dynamic population of monitoring agents is managed by our agile computing framework according to policies that define computing and networking resource restrictions as well as user notification requirements and preferences.

1 Introduction

The Florida Institute for Human and Machine Cognition (IHMC) is conducting research and development for automated safety and health monitoring of industrial chemical processes [1] as well as the NASA space shuttle fueling and launch process. Our current KAoS Reactive Monitoring and Event Notification (KARMEN) multi-agent system enables users to perform automated live monitoring of complex process conditions that were typically only detected manually or during post-analysis of recorded process data [2]. This research also extends to the complementary area of remote user notification, most notably concerning adapting the notification mode and salience based on the event context.

We have taken a human-centered approach to monitoring automation that complements the user's ability to identify relevant monitoring contexts with the software agent's ability to rapidly and vigilantly assess the process state. This frees our agents from needing complete and accurate models of the systems being monitored and also distinguishes our work from related multi-agent approaches to chemical and manufacturing systems diagnostics [3-5].

A brief description of KARMEN is followed by more detailed coverage of recent results involving how users describe process conditions to agents, how the resulting ad-hoc population of agents are deployed and managed by the framework, and the role of ontologies and policies in creating and controlling these agents.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 197 – 206, 2005.

2 KARMEN Overview

The health monitoring and process control of a complex system involves an extensive network of sensors, processors, and actuators. The elements of this process intranet may be linked together wirelessly or by more conventional means. In either case, unique opportunities for process control and health monitoring are offered by software agents that can migrate within the network to accomplish tasks specified by the human operators. Conceptually, a number of collaborating agents seek, collect, and evaluate data from individual sensors, interacting with other agents to form a composite picture of system state, and interacting with the human operators to provide information that is critical for system safety. The mobility of such agents (their ability to migrate within the system as required to accomplish tasks) introduces a previously unavailable degree of flexibility in the development of safety and health monitoring systems.

From the users' perspective, KARMEN is a desktop application for creating and deploying software agents to continuously monitor a process and notify the user as the conditions obtain and abate. Agent creation begins with condition specification. The user browses the space of sensors, valves, and other control elements comprising the process and selecting among the data streams each component provides.

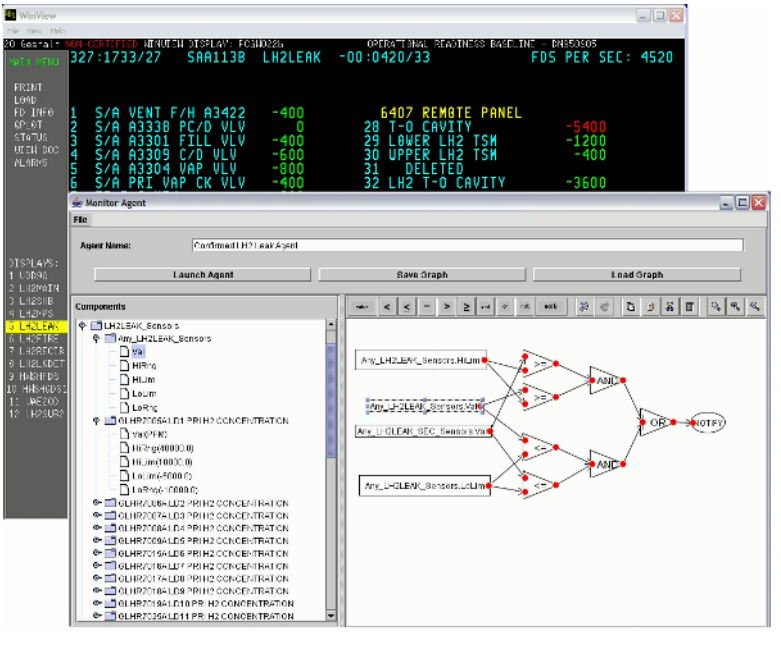


Fig. 1. Users construct graphical expressions to describe process conditions to monitoring agents, in this case involving the set of values from the operator screen shown in the background

$$(P_1 \geq P_{hi} \,\&\, S_1 \geq S_{hi}) \mid (P_1 \leq P_{low} \,\&\, S_1 \leq S_{low}) \mid (P_2 \geq P_{hi} \,\&\, S_2 \geq S_{hi}) \mid \ldots \quad (1)$$

Part of our research includes applying semantic web techniques to provide rich and flexible descriptions and classifications of sensors through both enumeration and common properties such as type, limits, capabilities, location, and status. The user next selects the control elements to monitor, then uses a graph-based interface to assemble the process variables into a logical expression. For example, liquid hydrogen (LH2) leak sensors are placed at intervals along a shuttle fueling line in pairs of primary and secondary (SEC) detectors. In this case there are over 50 sensors to monitor that are individually very sensitive and therefore prone to false positives. The graph in figure 1 expresses the condition that the high or low limit has been exceeded for a pair of primary and secondary LH2 leak sensors concurrently. The same expression is also depicted in formula 1 below. The user launches the agent into the KARMEN system where the agent survives indefinitely monitoring the process conditions and notifying the user. Perhaps through the KARMEN application if it is running, otherwise through email, text pager, or other modes.

From a systems perspective, KARMEN is a set of host environments configured to run these agents as well as a collection of services for managing them and the resources they consume. When a new agent is requested a central coordinator service determines where the agent is deployed based on the components referenced, the monitored condition expression and its sub-expressions, as well as available host resources and overlap with existing agents. The coordinator service establishes data feeds among agents and can move agents from host to host to balance processing and network load as well as keeping the agents running. Policies play an important role in restricting user access to sets of components, setting constraints on the resources agents are allowed to consume, and defining preferences concerning how the notification service will deliver messages.

3 KARMEN Architecture

KARMEN relies upon multi-agent frameworks developed at IHMC to manage and control its Java agent population as shown in figure 2. The FlexFeed agile computing framework provides mobile agent hosting, networking, and coordination. The KAoS policy framework provides tools and services for defining polices that constrain agent actions and resource usage as well as oblige agent actions including user notification. KARMEN and KAoS both rely upon semantic web ontologies developed using the W3C standard Web Ontology Language (OWL). KAoS also provides an extensible base ontology with services to query the ontology.

3.1 FlexFeed Agent Networking Framework

FlexFeed is a Java agent framework that facilitates communication between agents and manages the computing and network resources within a distributed multi-agent system [6, 7]. KARMEN agents rely on the FlexFeed API for mobile deployment and access to information feeds among heterogeneous sensor, intermediate, and user

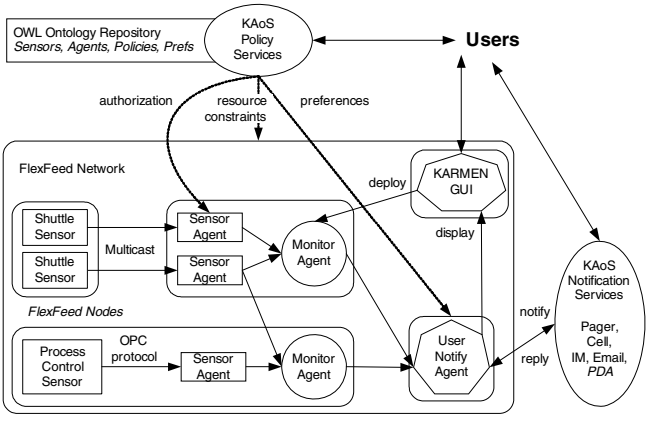


Fig. 2. A high-level depiction of the relationships among KARMEN system components

nodes. FlexFeed supports policies that restrict communication among agents and limit agent resource usage. The transport mechanism, message distribution, and filtering are each handled at the framework level, hiding these implementation details from the data producers and consumers. This architecture allows the framework to transparently customize the routing and transformation data streams while abstracting from the agent the tasks associated with the protocol selection, policies, and load balancing. Multiple communication protocols and lookup services can coexist in the network and FlexFeed will determine what protocols to use in order to distribute messages between any two nodes. This API provides two main components: the FlexFeed Manager, that handles agent lookup and delivery of data, and the FlexFeed Coordinator which is the intelligent component that is responsible for establishing and maintaining data streams in the framework. The Coordinator distributes processing load and bandwidth consumption across the framework preserving the resources on the nodes. Upon multiple requests on the same sensor, the Coordinator has the ability to discover and use intermediate processing nodes to minimize the network bandwidth and improve load balancing.

3.2 KAoS Policy Services

KAoS is a collection of policy services compatible with several software agent and robotic frameworks, as well as traditional distributed services platforms (e.g., CORBA, Web Services, Grid Computing) [8-10]. In the context of KARMEN, KAoS is used to define, manage, deconflict, and enforce policies restricting agent access to sensor data, bounding agent resources, and governing the mode of notification to users. The KAoS Policy Ontologies (KPO; http://ontology.ihmc.us/) are represented in the W3C standard Web Ontology Language (OWL) [11]. KAoS relies on an integrated theorem prover along with KAoS-specific extensions to support representation and reasoning about policies.

The current version of KPO defines basic ontologies for actions, actors, groups, places, various entities related to actions (e.g., computing resources), and policies. As

the application runs, classes and individuals corresponding to new policies and instances of application entities are also transparently added and deleted as needed. Through various property restrictions, a given policy can be appropriately scoped to various domains, for example, either to individual agents, to agents of a given class, to agents belonging to a particular group, or to agents running in a given physical place or computational environment. Additional aspects of the action context can be precisely described by restricting values of its properties. Groups of people, agents, and resources are also structured into ontologies to facilitate policy administration.

3.3 OWL Ontology Representation and Reasoning

Our system employs OWL to organize and classify process components, monitoring states, notification modes and salience, as well as users and organizational roles. OWL is a powerful description logic-based language developed for the semantic web. It provides vocabulary for describing properties and classes including relations between classes (e.g. disjointness), cardinality (e.g. “exactly one”), equality, rich typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. Combined with the reasoning capability of Stanford’s Java Theorem Prover (JTP; http://www.ksl.stanford.edu/software/jtp/), these ontologies enable users to effectively describe sophisticated monitoring conditions in a way that is accessible to agents. To make the use of OWL simple to non-specialist users, a number of graphical user interfaces have been defined.

We employ ontological classifications of process control components and events for expressing potentially large and complex aggregate monitoring conditions. to dynamically define custom limits and alarm conditions that were previously only pre-defined in the control system.

4 Process Monitoring

Some key challenges in monitoring automation include enabling users to easily describe conditions of interest to the monitoring software, allowing users to dynamically change the conditions being monitored without affecting the process control, automatically changing the monitored conditions in response to changes in the process state, efficiently evaluating the system state for the given conditions, and effectively communicating the process state back to the user.

4.1 Describing Process Conditions

KARMEN users define process conditions for agents using a graph-based tool to build expressions concerning process state as shown in figure 2. Users browse for individual sensors or classes of sensors and add these inputs to the graph. Nodes are then selected to compare, combine, and transform these sensor inputs into a logical expression. When the user launches the agent, each sub-expression can be assigned to an existing agent in the FlexFeed network for evaluation or new agents be created as needed.

One particularly valuable aspect of the research involves enabling users to monitor complex and aggregate process conditions that could not previously be monitored at

runtime. Defining ontologies of process variables in OWL enables users to organize and classify sensors by relevant properties to easily express complex monitoring conditions for groups of related sensors (e.g. monitor for any sensor value from shuttle main engine one that exceeds 90% of its associated high alarm limit). Using ontological classes in monitoring expressions allows users to define complex aggregate conditions concisely. For example, the class of "all sensors on main engine one with a high limit value" can be constructed in the ontology based on the common properties of individual sensors such as location and limits. Such an ontology class can then be incorporated into a monitoring expression such as "sensor current value greater than 90% of sensor's high limit". This allows users to define conditions at a variety of scopes from the very narrow and specific to system-wide.

4.2 Monitoring Capabilities

The most basic capabilities of the process monitoring agents for this system include comparing process variables to scalar values and other process variables (e.g. monitor for a valve's actuator position greater than its predefined high alarm limit). We effectively extend the alarm functionality commercial control systems provide with the added value of making this capability available for ad-hoc and remote use. The ability to inject new conditions non-intrusively into an operational environment is critical. We can further incorporate monitoring statistical summaries of sensor behavior including standard deviation, variance, mean, and rate of change over a given time period or number of samples. Users can also employ mathematical expressions to derive new aggregate conditions (e.g., monitor the product of pressure and temperature sensor readings), annotate process variables such as defining progressive high and low limits, and access system annotations such as maximum, minimum, and average observed values from historical data. These feature support live, flexible monitoring using new combinations of parameters not inherent in the control system.

Adding remote monitoring capabilities carries the responsibility to control access to sensitive data. The KAoS policy services leverage the ontologies defined for classifying sensors to also define and enforce authorization policies that restrict access to process data such as "IHMC personnel can only access sensors in the shuttle main engine class" which will deny authorization to access feeds from these sensors to all agents created by IHMC personnel. Such policies could also describe reductions of sampling rate or precision which the agents would enforce.

4.3 User Notification

Notifications are generated as the monitored conditions obtain and abate. The mode, salience, and recipients of each notification are governed by KAoS policies representing organizational requirements and users' personal preferences. Notification modes may include E-mail, Instant Message, Pager, and Operator Displays. Notification policies typically cover such factors as event type, severity, the recipient's organizational role and presence, and the plant area in which the event occurred. For example, a policy might be to "page an onsite Field Operator immediately when a critical H2 Plant monitored condition is satisfied and the Process

Engineer is unavailable". The selection of mode, recipients, and salience is made at runtime based on information gathered about the user's presence and the modes available (e.g. the user's instant message client indicates user is available and the user's schedule indicates she is onsite).

The default behavior of the Notification agent is to display messages in the KARMEN application interface. All other notification actions are governed by KAoS policies representing organizational requirements and personal preferences. Each policy obliges the notification agent to take certain actions based on the qualities of the monitored event and the current disposition of the concerned personnel. We have developed a set of initial ontologies depicted in figure 3 for notification that draws heavily on the work of Schrekenghost and colleagues [13].

The current event characteristics that can trigger a policy include the event type (satisfied/unsatisfied condition, activated/deactivated alarm: see ConditionStatus in figure 3), the assigned event severity (critical, warning, advisory, log), and the plant area in which the event occurred based on the component hierarchy defined in the ontology.

The user characteristics that can trigger a policy include the user's organizational role (operator, process engineer, area manager, etc.) and the user's current physical and computational presence (nearby/remote, online/offline: see figure 3). The qualities of the notification action that policies can oblige include the mode, latency, and focus of attention. Notification modes currently include e-mail, instant message, pager, operator displays, and the IHMC Monitor application. The latency quality controls how quickly the user is notified (immediate, deferred, archive). The focus of attention quality controls how forcefully the user's attention is obtained and depends on the features available in each notification mode (e.g. instant message chat session that interrupts the user vs. a queued message in the background).

The notification obligation policies are created using the KAoS Policy Administration Tool (KPAT) shown in figure 4. The attributes of the policy are from the ontological concepts shown in figure 3.

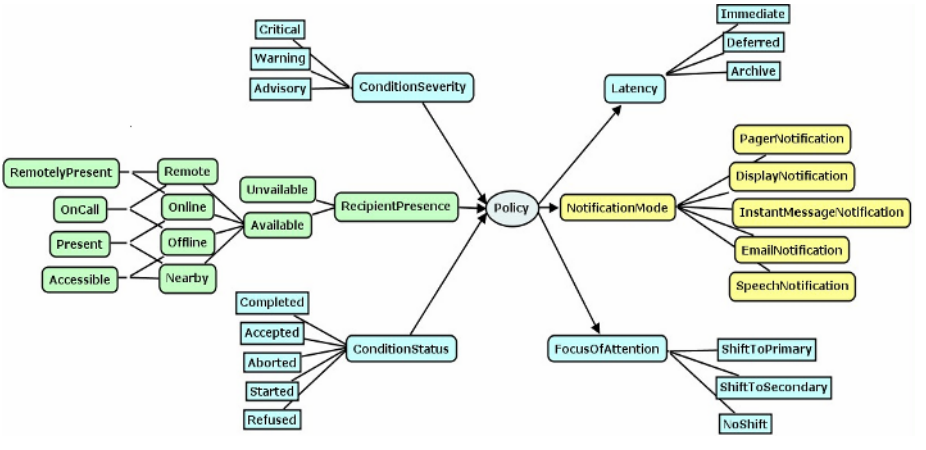


Fig. 3. OWL ontologies used by the KARMEN system for notification are graphically depicted

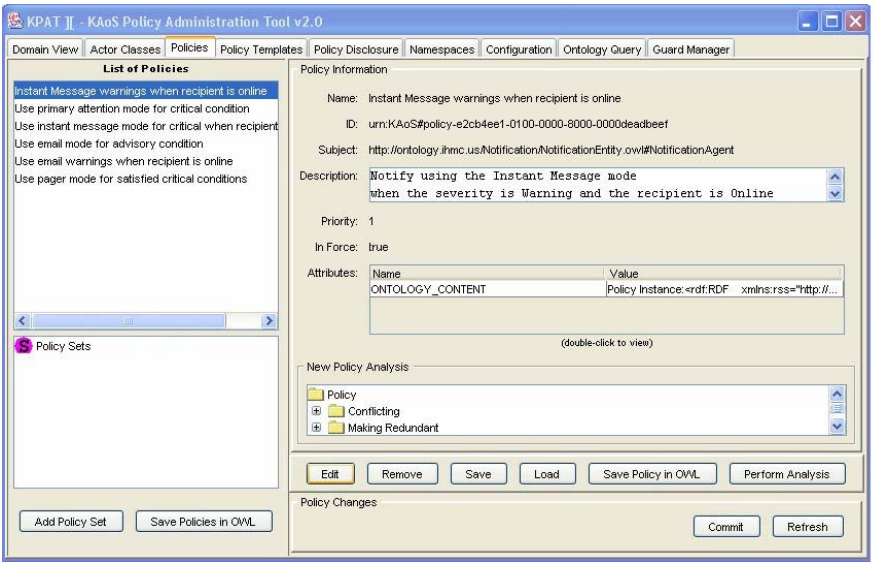


Fig. 4. The KAoS Policy Administration Tool (KPAT) screen displaying a sample set of policies that govern notification modes in the KARMEN system

Multiple policies can apply to a single event such as using the pager mode for critical events, using the instant message mode for critical events when the user is online, and using the primary focus of attention for critical events. Each policy is assigned a priority. KAoS uses the priority to resolve policy conflicts thereby enforcing organizational policies over personal preferences. The user notification agents can require and obtain acknowledgement of notifications and escalate the notification mode and recipients when acknowledgement is not received in a specified timeframe. In the near future, agents will select notification mode and salience based on the recipient's responsiveness to previous notification attempts by learning the most effective mode of contacting each user according to the time, user presence, and condition severity. Monitoring agents can begin recording a set of sensor values when the specified conditions obtain, stop recording when the conditions abate (or after a certain duration), then include a graph of the recorded data as an attachment to electronic notifications. Summaries could be extended to several formats including movies, spreadsheets, and PDF files.

5 Summary

The KARMEN system is distinguished by its human-centered approach of providing tools to create personal monitoring agents with rich semantic descriptions of process state and salient user notification. These agent-based tools complement and amplify the expertise of the engineers and operators with the ability to create and refine personally relevant assessments of live process conditions. KARMEN focuses on

supporting users in the difficult task of safely and effectively operating complex processes. We enable operators to specify complex monitoring conditions by using intuitive graphical tools; these conditions can be changed at any time without affecting the process control. Users also can apply ontological classes to define complex aggregate conditions that have not been previously specified in real-time.

References

1. Bunch, L., Breedy, M., Bradshaw, J., Carvalho, M., Suri, N., Uszok, A., Hansen, J., Pechoucek, M., and Marik, V. 2004. Software Agents for Process Monitoring and Notification. In Proceedings of the ACM Symposium for Applied Computing, Nicosia, Cyprus, 94-99. New York: ACM.
2. T. Blevins, G. McMillan, W. Wijsznis, and M. Brown, *Advanced Control Unleashed: Plant Performance Management for Optimum Benefit*. Research Triangle Park, NC: The Instrumentation, Systems, and Automation Society, 2003, pp. 163-182.
3. N. Hamdy and R. Fulvio, "Abnormal Condition Management with Real-time Expert System and Object Technology," *PCAI*, vol 17(1), pp. 28-35, 2003.
4. F. Heck, T. Laengle, H. Woern, "A Multi-Agent Based monitoring and Diagnosis System for Industrial Components," University of Karlsruhe, Institute for Process Control and Robotics. Karlsruhe, Germany.
5. I.A. Letia, F.Craciun, Z. Kope, A. Netin, "Distibuted Diagnosis by BDI Agents". In Proceedings of the IASTED International Conference APPLIED INFORMATICS. Innsbruck, Austria. 2000.
6. M. Carvalho and M. Breedy, "Supporting Flexible Data Feeds in Dynamic Sensor Grids Through Mobile Agents". In *Proceedings of the 6th International Conference in Mobile Agents*, Barcelona, Spain, October 2002.
7. N. Suri, J.M. Bradshaw, M. Breedy, P. Groth, G. Hill, R. Jeffers, and T. Mitrovich, "An Overview of the NOMADS Mobile Agent System," Sixth ECOOP Workshop on Mobile Object System. Available: http://cui.unige.ch/~ecoopws/ws00.
8. A. Uszok, J.M. Bradshaw, R. Jeffers, N. Suri, P. Hayes M Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, "KAoS policy and Domain Services: Toward a Description-logic Approach to Policy Representation, Deconfliction, and Enforcement," In Proceedings of IEEE Fourth International Workshop on Policy. Lake Como, Italy, June 2003, pp. 93-98.
9. J.M. Bradshaw, A. Uszok, R. Jeffers, N. Suri P. Hayes, M. Burstein, A. Acquisti,, B. Benyo, M. Breedy, M. Carvalho, D. Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, & R. Van Hoof, "Representation and Reasoning for DAML-based Policy and Domain Services in KAoS and Nomads," In Proceedings of the Autonomous Agents and Multi-Agent Systems Conference. Melbourne, Australia. ACM Press, New York, NY, 2003, pp. 835-842.
10. J.M. Bradshaw, P. Beautement, M. Breedy, L. Bunch, S. Drakunov, P.J. Feltovich, R.R. Hoffman, R. Jeffers, M. Johnson, S. Kulkarni, J. Lott, A. Raj, N. Suri, & A. Uszok, "Making Agents Acceptable to People," In Intelligent Technologies for Information Analysis: Advances in Agents, Data Mining, and Statistical Learning, N. Zhong and J. Liu, Eds. Berlin, Germany, Springer Verlag, 2004, pp. 361-400.
11. J. Hendler, T. Berners-Lee and E. Miller, "Integrating Applications on the Semantic Web," Journal of the Institute of Electrical Engineers of Japan, vol 122(10), October, 2002, pp. 676-680.

12. D. Schreckenghost, C. Martin, C. Thronesbery, “Specifying Organizational Policies and Individual Preferences for Human-Software Interaction,” In Etiquette for Human-Computer Work, Papers from the AAAI Fall Symposium. Technical Report FS-02-02, AAAI Press, 2003.
13. C. Glymour, K. McGlaughlin, “Analyzing A Data Lookup Method for Machine Learning in Monitoring and Fault Localization for Hydrogen Generation Plants, Chemical Processing Plants and Other Complex Systems,” Final Report for UCF contract 26-56-208. Pensacola, FL, September 2003.
14. S.D.J. McArthur, E.M. Davidson, J.A. Hossack and J.R. Mc Donald. Automating Power System Fault Diagnosis through Multi-Agent System Technology. In Proceedings of the Hawaii International Conference on System Sciences. 2004.

Simulation of Underwater Surveillance by a Team of Autonomous Robots

Milan Rollo[1], Petr Novák[1], and Pavel Jisl[2]

[1] Center of Applied Cybernetics, Czech Technical University in Prague
[2] Gerstner Laboratory, Czech Technical University in Prague,
Technická 2, Prague 6, 166 27 Czech Republic
{rollo, novakpe, jisl}@labe.felk.cvut.cz

Abstract. Within this paper we describe a simulation environment for the underwater surveillance and propose architecture of control part of autonomous robot capable of efficient operation in such environment. Besides this the algorithms for decentralized coordination within a group of such robots and video stream transmission path planning are discussed. Development of these algorithms was necessary due to the nature of environment, where individual robots can become temporarily inaccessible.

1 Introduction

Real-life experimentation can be costly, take a long time or be even impossible due to the physical or security reasons. One of the ways how to deal with this problem is to create a model of the real-life process or environment and use a software simulation. Multi-agent systems are a natural choice to model distributed systems consisting of autonomous, self-interested entities like the teams of autonomous robots. Such a model can be used with advantages to simulate and analyze specific aspects of the system's overall behavior like the activity coordination, negotiation or mutual communication. Beside this the model can also be used to develop and test control algorithms before their practical deployment.

Goal of the project we describe in this paper was to develop a simulation environment and model of the underwater surveillance exercise and propose architecture of autonomous robots capable of efficient and robust collaboration in such environment. In our scenario the goal of the group of autonomous robots (unmanned underwater vehicles - UUVs) is to search a given coast area, detect and remove all mines located there. To allow mine removal a video transmission path must be established between the base (operated by human crew that gives robot a permission to remove mine) and the robot who has found the mine. Typically, relying via other robots is necessary [1], because the video transmission range is limited (due to the necessity to use acoustic modems in underwater environment). Problem of underwater image and data transfer is described e.g. in [2].

In such a type of scenario no dedicated central planning entity can be used. There are two main reasons. Because of the limited communication accessibility, robots can easy get out of the central entity's communication range. Robots

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 207–220, 2005.

thus will have to return back within the communication range to get new commands (tasks) and to update central entity's knowledge about the environment (necessary for accurate and efficient planning). Second reason is that in case of malfunction of this entity robots without their own planning capabilities will fail to operate efficiently and coordinate their actions. All robots in this scenario are thus autonomous and independent, each of them can take a role of the task coordinator. Robots communicate and negotiate in peer-to-peer manner when fulfilling the tasks. Unlike the other works dealing with teams of autonomous robots, where each robot has different capabilities and their authors investigate mainly the team action planning activities e.g. [3], [4], in this project we focused on communication and knowledge synchronization in environment with partial communication inaccessibility and transmission path establishment algorithms. We also do not deal with the problem of individual robot's navigation and localization (see e.g. [5], [6]).

Due to the above mentioned scenario features we decided to use $\mathcal{A}$-**globe** multi-agent platform as a simulation environment. Beside the functions common to most of agent platforms (like agent life-cycle control, message transport, etc.) $\mathcal{A}$-**globe** provides user with agent mobility, communication inaccessibility and geographical simulation support.

The paper is organized as follows. Internal simulation support of the $\mathcal{A}$-**globe** agent platform is described in the next section. Section 3 concerns on the problem of environment simulation and architecture of robot's control part. Different methods and algorithms for transmission collaborator search and transmission path planning are described in section 4. Experiments carried out and deployment on hardware robots are described in sections 5 and 6 respectively.

2 Simulation Support in $\mathcal{A}$-globe Multi-agent Platform

$\mathcal{A}$-**globe** is suitable for real-world simulations including both static and mobile units (e.g. logistics, ad-hoc networking simulation), where the core platform is extended by a set of services provided by Geographical Information System (GIS) and Environment Simulator (ES) agent. The ES agent simulates dynamics (physical location, movement in time and others parameters) of each unit.

Simulation scenario is defined by a set of actors represented by agents residing in the agent containers. All agent containers are connected together to one system by the GIS services. Beside the simulation of dynamics the ES agent can also control communication accessibility among all agent containers. The GIS service applies accessibility restrictions in the message transport layer of the agent container.

The agents used for the world simulation are all located in a dedicated master container and are called *Environment Simulation* agents. These agents only rarely use messages to communicate with actor agents. Instead, they communicate with *topic messaging* - container-to-container messaging specifically reserved for environmental simulation. Topic messaging is managed by GIS Service - a special service that can be started in individual containers. Client

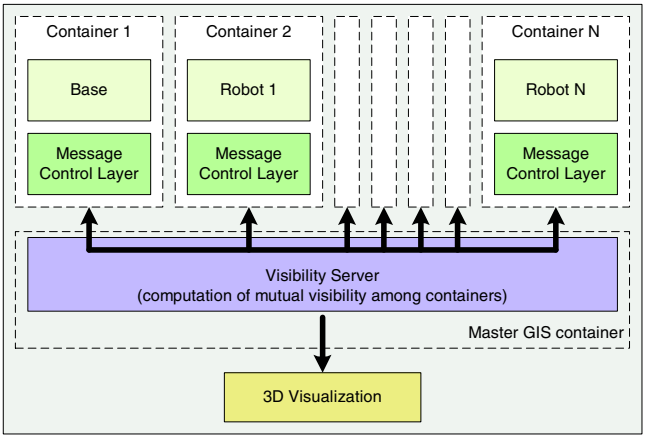


Fig. 1. Underwater Surveillance Scenario Architecture

agents (actors) subscribe with GIS client service to receive various topics. In addition, the agents who wish to act on the environment can submit topics to the GIS service. These topics are then sent to all ES agents in the master container subscribed to receive the topic.

For more detailed description of the $\mathcal{A}$-**globe** system architecture and its comparison with other agent platforms see [7], platform is freely available at [8].

3 Underwater Surveillance Scenario Implementation

In this scenario we use the above mentioned agent platform to simulate a real-world environment where group of collaborative, autonomous robots searches a coast area, detects and removes all mines located there.

To attend mainly to the algorithms of distributed coordination and transmission path planning we have adopted some simplifications:

- We do not cope with exact simulation of robot's sensors and actuators. Additional simulation tools (e.g. Gazebo or Stage [9]) can be used for this purpose.
- The coast area is flat without any obstacles preventing from movement or transmission.
- Robots are able to stay on position without movement (have ability to float).
- Robots have unlimited energy for movement and transmission.
- We assume that robots know their geographical positions at any moment (we suppose presence of some GPS-like positioning device or localization based on the processing of data from sensors).

Each of the robots is represented by one $\mathcal{A}$-**globe** agent container. Containers are connected together into one system by the GIS services. Visibility Server that

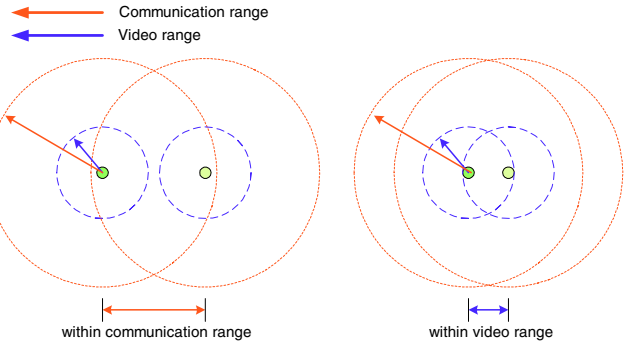


Fig. 2. Transmission Range

computes the mutual communication accessibility among containers is running on the master container. Scenario architecture is in figure 1.

Two types of communication accessibility are simulated in this scenario (see figure 2) – **high bandwidth** accessibility, necessary for video transmissions (very restrained) and **signaling** accessibility, used for coordination messages and position information (higher than video accessibility, but remains limited).

Visibility Server informs robots not only about the other accessible robots, but the information is extended with additional attribute representing the degree of signal strength. To allow a video transmission, certain degree must be reached. Each robot consists of several components, implemented as $\mathcal{A}$**-globe** agents running within one agent container (figure 3):

- **Robot Pod** simulator, computes robot's moves and updates its position with GIS server via GIS service. Robot Pod receives commands from the Coordinator Agent.
- **Mine Detector** simulator, provides the decision-making components with information about found mines. It informs the Coordinator Agent about all suspicions objects within the detection range.
- **Video** stream acquisition and transmission element. This subsystem creates the data feed from the source provided by the simulation and prepares transmission path by remotely spawning one-use transmission agents along the path. Video is then transmitted as a stream of binary encoded messages.
- **Robot Coordinator** implements search algorithm, transmission coalition establishment and negotiation.

3.1 Internal Structure of the Coordinator Agent

Coordinator Agent represents the control unit of the robot. It consists of several internal modules, as shown in figure 4:

Control Unit: Control unit is responsible for operation of the Coordinator Agent. It communicates with the RobotPod Agent and Detection Agent

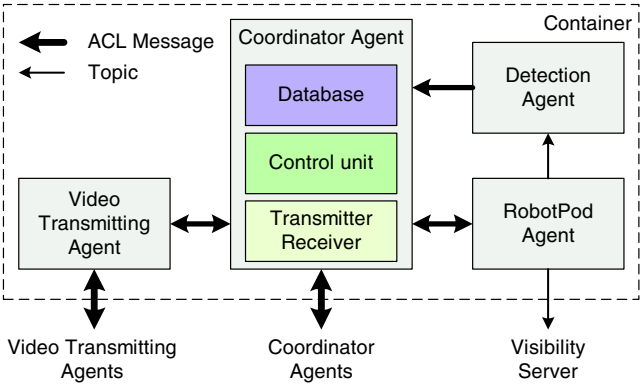


Fig. 3. Robot's Internal Structure

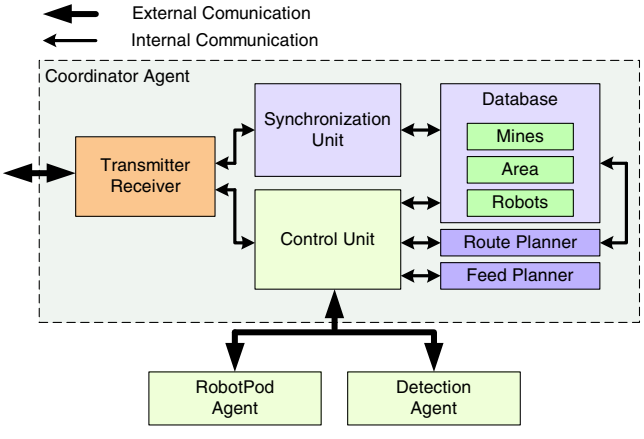


Fig. 4. Coordinator Agent's Architecture

present on the local container and also with the Coordinator Agents on remote robots. Local communication is managed by the agent platform via ACL messages, communication with other Coordinators is realized by the Transmitter/Receiver unit.

Transmitter/Receiver: This unit simulates the communication restrictions of the real-life transmitters used on the robots (acoustic modems in this case). Each message the Coordinator wants to send to another Coordinator (including the video transmission messages) is first put into the queue of outgoing messages. Transmitter unit withdraws messages from this queue on the FIFO basis, sends the messages and blocks the communication channel for a certain period of time. This way it simulates the time the transmission of the previous message will go on. Length of the time delay between the messages depends on the size of the message, capacity of the communication channel and distance of the receiver.

Database Module: This module maintains the Coordinator's knowledge about the mines, searched area and other robots. Information in this database can be updated either by the robot's own observations or by synchronization messages received from other agents. Direct observations include change of the mine status (new mine found, video transmission finished, off-line video recorded), change of the area status (new part of the area to be searched was assigned, part of the area was searched) and change of the communication accessibility with other robots. In case of mine or area status change Coordinator tries to propagate the information among other Coordinators using the synchronization unit.

Synchronization Unit: This unit is responsible for keeping the Database Module updated and consistent with other robots. Two robots mutually synchronize their databases either when they get within the communication accessibility range or if one of them updated its own database (can be caused by the direct observation or by receiving a synchronization message from the third robot). Each Coordinator maintains the record with the time of last synchronization with other robots. This record is used when the communication accessibility link is restored - Coordinators exchange synchronization messages that contain database records with time stamp later than the last synchronization time. Synchronization Unit is also responsible for solving possible conflicts between the actual database records and received synchronization messages. Each Coordinator Agent provides user with a graphical representation of its internal database record.

Route Planner: This unit plans agent's route through the area. Result of the planning algorithm is the list of waypoints robot has to reach to search its part of area. Waypoints are selected as boustrophedon path that covers the area. In current version each robot has assigned its part of area statically at the beginning of search. New version is now under development, where robots dynamically assign smaller parts of the area using the auction techniques.

Transmission Path Planner: Once a new mine is found, this unit is responsible for selecting the best video transmission path according to the given optimization criteria. This unit is described in details in section 4.2.

4 Distributing the Coordination Process

The process of agents' coordination can be done centrally by a dedicated central coordination agent. This however makes the community fragile and dependent on a central coordination unit. At the same time this element may become a bottleneck in situations when several robots request new tasks simultaneously or are out of the entity's communication range. Alternatively, we can distribute the coordination processes in several ways:

level-1: There is no central coordination agent. Each robot (especially the robots that locate the suspicions object) can become a coordinator for a single feed planning process. The result of such a planning activity is a central plan that is imposed upon the involved robots.

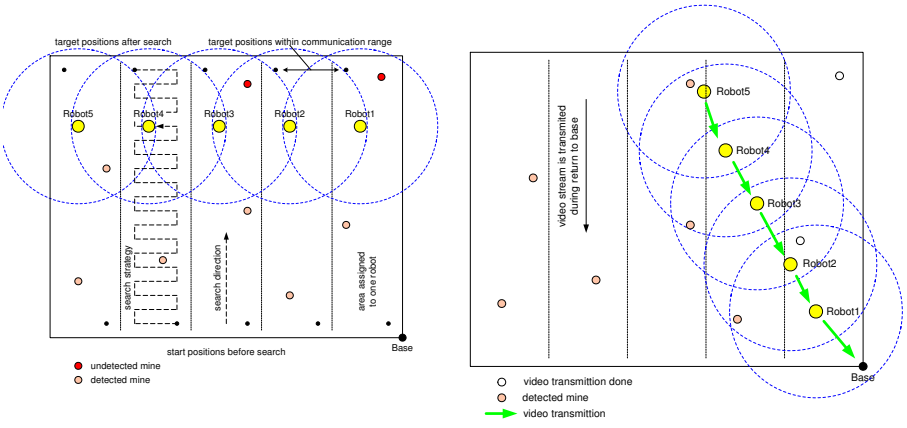


Fig. 5. Central Search and Transmission Planning Algorithm

level-2: The coordination process is in parts distributed among the agents. There is no central plan. The robots that are supposed to form the feed are selected by the agent who detects the mine. However each robot locates its desired position so that the feed is established.

level-3: The coordination process is distributed completely among the agents. There is no central plan. The feed-members selection process is distributed recursively. In this situation backtracking of robots through the area when searching for collaborators needs to be made possible.

The level-1 and level-2 distribution is desired for the increased efficiency, flexibility and survivability of the coordination process. The level-3 distribution of coordination makes sense only in the situations when it is impossible to bring all the planning information to the coordinator. This is very often the case of semi-trusted environment (not our case) or in the situations with very large area and small number of agents.

4.1 Transmission Collaborators Search Strategies

To transmit the video stream of suspicions object to human operator, relaying via several collaborators is usually required. Robot that finds a suspicions object is responsible for retrieval of enough collaborators to send the transmission online. If sufficient number of collaborators is not available, robot records video stream into its memory. Three different collaborator search strategies were tested:

Central Planning: Using this algorithm robots do not form the video feed immediately after the object is found, but store the object's position in memory and continue in search. After all robots search the assigned parts of area one of them gets information about all objects and makes a plan that is imposed upon other robots (all robots are within the communication range with its neighbors during the planning phase). Video stream of all objects is transmitted on their way back to base (see figure 5). This approach allows to

make an optimal plan (complete information is available) and it is ensured that video stream of all objects will be send online.

Relayed Collaborator Search: Robot who finds suspicions object becomes a coordinator of the transmission planning and immediately asks other robots within its communication for their actual status and position. These robots forward this request to their visible neighbors and so on (formation of loops is prevented). If robot has no other neighbors to forward the message to (or has received answers from all of them) it returns the desired information back to its predecessor. This way the collaborator collects information about all robots within its "relayed" communication range. If sufficient number of collaborators is found, coordinator uses this information to plan the transmission feed (see section 4.2).

Elastic Collaborator Search: In some cases robots can not find enough collaborators even using the relayed communication. This algorithm allows robot to leave the suspicions object and look for the missing collaborators (we suppose robot knows number of robots that are searching the area and theirs assigned parts). Group of all participating collaborators moves together in regular order (to remain within a communication range) and searches areas assigned to missing robots. Robots continue in search until the sufficient number of robots is found or whole area is unsuccessfully searched.

4.2 Transmission Path Planning

We have developed three different distributed coordination algorithms to build the ad-hoc data transmission feed. The most straightforward are the approaches relying on a single agent mastering the planning process. Upon finding the mine, it requests the other visible robots to move to a specific positions so that a high-bandwidth transmission link between the mine and the base is established. In two variants of this approach, we may emphasize either the communication quality or the minimization of other robots' actions disturbance. When we optimize the communication quality, relay robots tend to be placed on the join between the mine and the base so that the overall distance between the relays is minimized and minimal possible number of robots is used. On the other hand, when we try to minimize the impact on relay robots' own plans, relays are spread in the area between the transmission origin and target, in the proximity of their original areas. In the third approach, the control over the feed planning is not centralized, but rather passed along the communication link relays when the connection is constructed. This approach is well adopted for the environments where the communication is limited and the knowledge necessary for feed building is not common, but rather distributed among robots.

Direct Line Transmission Path Planner (DLTP): This is the simplest planner implemented within the project. It achieves the level-1 coordination process distribution. Robot who finds a suspicions object becomes a coordinator of the transmission planning and asks other robots for their status. Its goal is then to select the best subset from all robots that agreed to participate on the

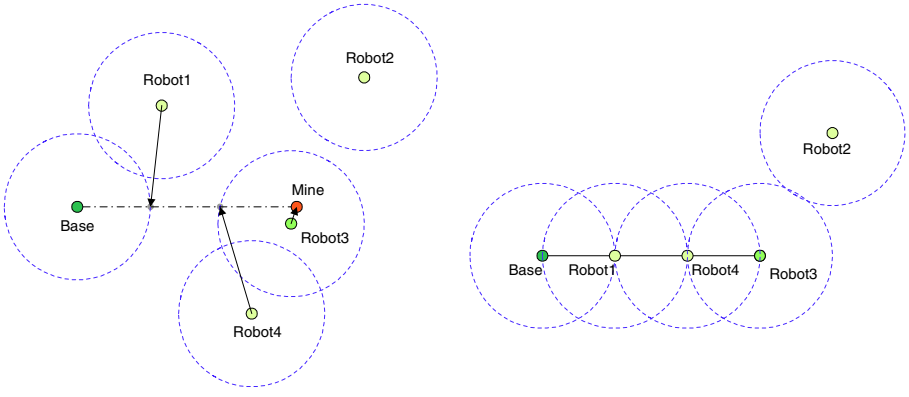


Fig. 6. Direct Line Transmission Path Planner: (a) – Original placement of robots; (b) – Robots on the video transmission positions.

transmission. Participants are placed on the join of the Base and mine position in periodic distances (equal to the video transmission range), see figure 6. Minimal necessary number of participants is:

$$n = \text{ceil}\left(\frac{\text{distance}(Base, mine)}{videoRange}\right) - 1.$$

Number of participants is used as length of variation for evaluating sets of possible participants. These sets are computed as variations from all available robots with length n. The optimization function is evaluated for all sets and the best set is chosen. Optimization criterion minimizes robots' total moving distance from theirs original positions to transmission positions (minimizes the energy consumption):

$$\min\left(\sum_{i=1}^{n} \text{distance}\left(intersection_i, robot\right)\right), \forall setRobots_{\text{card}=n} \subseteq Available$$

Minimal Time-To-Transmit Planner (M3TP): Using the Minimal Time-To-Transmit Planner more than the minimal required number of robots can participate in video transmission (in contrast to the DLTP). Optimization criterion is in this case to minimize the time intermediate robots spend on transmission:

$$\min\left(\sum_{i=1}^{n} moveTime_i + n.transmitTime + n.\max\left(moveTime\right)\right)$$

$$\forall setRobots \subseteq Available$$

Where:

- $Available$ is set of all robots that agreed to participate on video transmission;
- $transmitTime$ represents the estimation of time the transmission will go on (real value depends on the operator's decision);

- n is cardinality of the set $setRobots$ ($n \geq$ minimal necessary number of participants);
- $moveTime$ is the time it takes to the robot to move from its original position to the transmission position.

Problem when solving this optimization function is that we do not know the optimal position the transmitting robots should occupy. Optimal placement of the robots can form a general curve not only a line as in the previous case. It is computationally infeasible to search the whole state space (i.e. test all possible placements of all sets of robots) – new algorithm was thus developed. This algorithm works in two phases:

(i) Selection of the best subset of robots from all available: Instead of measuring the move time to transmission position, time to get within the video transmission range of the previous robot is measured (we are looking for maximally continuous path in graph). Each robot has assigned an initial value representing the shortest path to this node through the minimal number of robots required to build the feed. Modified Dijkstras graph search algorithm is then used to find the shortest path to base. This modification deals with the maximum move time which affects all robots on the path, not only an actual arc. Second issue is the transmission time, which can differ for each robot and depends also on its actual state.

(ii) Computation of individual robots' positions: Once the best subset of robots is selected, theirs placement in transmission feed is computed. Starting from the video stream source, each robot's new position is selected as either the intersection of join of previous and next robot with the previous robot's video visibility range circle or the closest point from which both the previous and next robots are reachable. If the computed position of the last robot is not within the video communication range of the base, we change the direction of computation (the direction is now from the base to the video source). Algorithm works in loop until all robots can see the previous and next robot in feed. This process is shown schematically in figure 7.

Fully Decentralized Planner (FDP): In case of this transmission path planner the positions of the participating robots are not computed centrally. Robot that found a mine only verifies the accessibility of the minimal required number of robots. If such a number of robots is available, subset of all mutually accessible robots is selected to build a transmission path. Mutual accessibility is the necessary condition to coordinate the robots' movement.

All these robots are then informed about the order of the transmission path. Each of them starts to move on position where both the previous and next robot in the feed are accessible for the video transmission. If such a position does not exist, it moves to the midpoint of join of those two robots and informs them about its new positions during the movement. In consequence of this movement positions of the target points of the neighboring robots change dynamically (end points of the join whose midpoint the robot is trying to reach are moving).

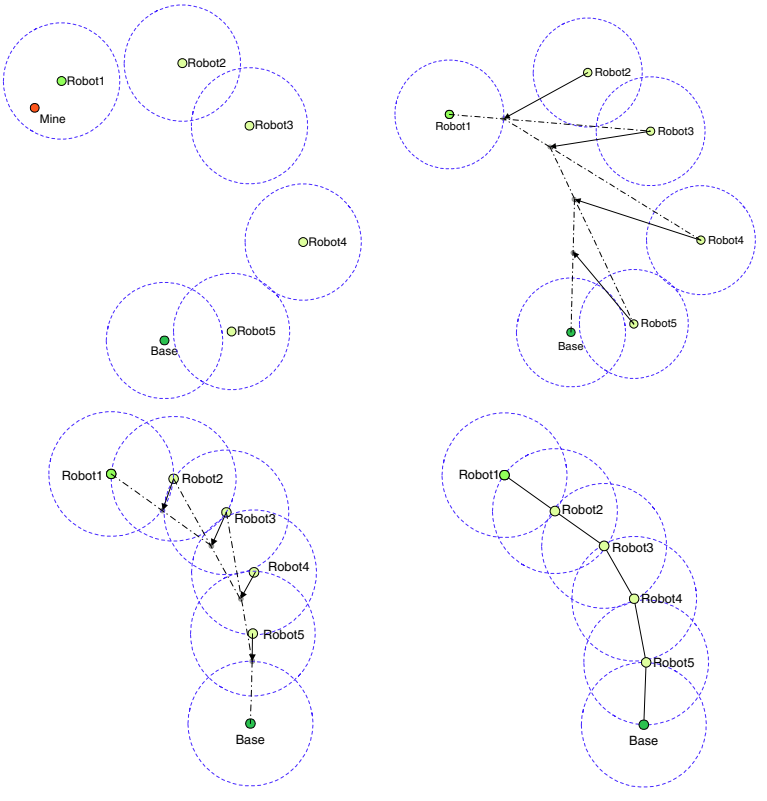


Fig. 7. Minimal Time-To-Transmit Planner (M3TP): process of the transmission robots' placement computation

Robot that found a mine periodically attempts to establish a video connection with base. Once this attempt is successful (all robots in the transmission path are within a video range) the Transmission Agents deploy on robots and video transmission begins.

5 Experiments

A set of experiments was carried out in order to study especially the features of transmission collaborators search strategies. We were using two different environment setups: (i) mines were placed in pattern (covering the extreme positions), (ii) mines were placed randomly with uniform distribution.

Simulated area was a square with side length 1 km, base is placed in a corner. We assume that robots are equipped with acoustic modems with high bandwidth communication range 300 m. To allow video transmission from farthermost place (corner diagonally from base), area has to be searched by at least five robots.

As shown in fig. 8 use of the central planning algorithm ensures that all video streams will be transmitted online (algorithm is independent on the signaling

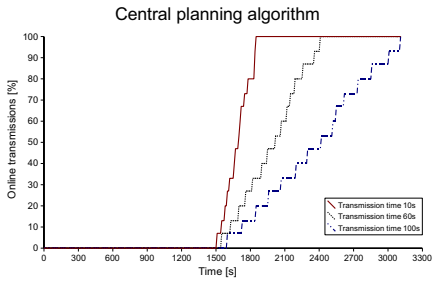


Fig. 8. Central planning algorithm - dependency of the number of online transmissions on length of transmission time

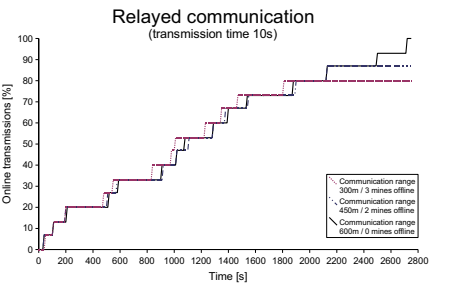


Fig. 9. Relayed communication - influence of communication range on the number of online transmissions

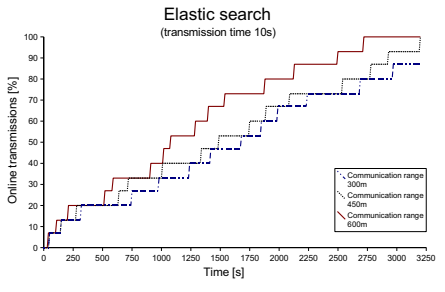


Fig. 10. Elastic search - influence of communication range on the number of online transmissions

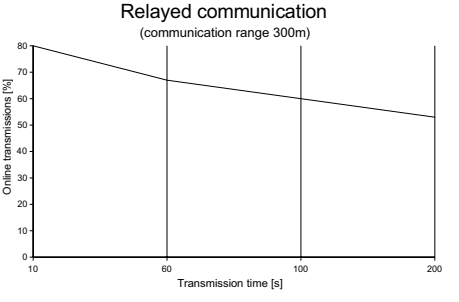


Fig. 11. Relayed communication - dependency of the number of online transmissions on length of transmission time

communication range). For short transmission times this algorithm performs best also for length of the overall area search time. However for a long transmission times and slow movement speeds the relayed search algorithm can achieve better overall times - robots transmit during the area search phase and rest of the robots can search their parts in the meantime. When transmitting during returning to base robots have to wait for others to finish their transmission and waste time.

Relayed search algorithm can be with advantages used in environments where each new detected object can bring additional tasks (e.g. special dedicated robots are send to remove the mines or perform detailed examination). This algorithm can be also interrupted at any moment and at least part of the area can be marked as searched (anytime algorithm). Disadvantage of the this algorithm is that with decreasing signaling communication ranges and increasing transmission times decreases the number of online transmissions (see figures 9, 11).

Use of elastic search algorithm increases the number of online transmissions compared to relayed search algorithm, but for the price of longer transmission times (fig. 10).

6 Deployment on Hardware Robots

Generality of the $\mathcal{A}$-**globe** technology and functionality of the coordination algorithms has been proved when migrating the technology to the robocup soccer environment. The GIS server and ES agents managing the position of the robots have been replaced by the information from the camera observing the robocup soccer field. Similarly the RobotPod agent has been directly coupled with the hardware of the robocup soccer robots. Screen shot from the experimental testing is in figure 12.

Fig. 12. Relayed communication as simulated by Robocup soccer robots. Robot movements and positions are derived from hardware inputs. Base is in the left corner of area.

7 Conclusion and Future Work

Within this project we have developed a specific simulation environment using the $\mathcal{A}$-**globe** multi-agent platform. Main reason to develop such environment was to enable a software simulation of real-life hardware robots where scalability experiments and efficient development and verification of embedded decision making algorithms can be carried out.

The experiments conducted in the simulation environment with various environment settings (movement speed, number of mines, length of the transmission time) have proved that each collaborator search strategy is suitable for different areas of tasks (e.g. hydrographic or geophysical surveys, minesweeping, etc.). Their pros and cons are discussed in the previous section. Based on the actual task and environment features, operator can decide which algorithm will be used. Features of the developed control architecture were verified by the set of experiments with deployment on hardware robots.

Besides carrying out some additional experiments and measuring the efficiency of suggested algorithms, we plan to further elaborate the methods of distributed coordination. We will also study appropriateness of the particular levels of distribution of the coordination processes. The fully decentralized planners (FDP) will be optimized and new algorithm for the level-3 coordination process will be developed. Methods for an efficient feed sharing and dynamic division of the area and will be further studied.

Acknowledgement

The project work has been in part co-funded by ONR project no.: N00014-03-1-0292 (Naval Automation and Information Management Technology) and by the Grant no.: 1M6840770004 of the Ministry of Education, Youth and Sports of the Czech Republic (Center of Applied Cybernetics).

References

1. Rehák, M., Pěchouček, M., Tožička, J., Šišlák, D.: Using stand-in agents in partially accessible multi-agent environment. In: Proceedings of Engineering Societies in the Agents World V, Toulouse, October 2004. Number 3451 in LNAI, Springer-Verlag, Heidelberg (2005) 277–291
2. Benthos Inc.: Transport of Underwater Images, Virtual Acquisition Showcase 2004. `http://www.dawnbreaker.com/virtual2004/briefings/Benthos.pdf` (2004)
3. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research **7** (1997) 83–124
4. Kaminka, G.A., Bowling, M.: Towards robust teams with many agents. Technical report, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213 (2001)
5. Whitcomb, L., Yoerger, D., Singh, H., Mindell, D.: Towards precision robotic maneuvering, survey, and manipulation in unstructured undersea environments. In: Robotics Research - The Eighth International Symposium, Springer-Verlag, London (1998)
6. Whitcomb, L., Yoerger, D., Singh, H., Howland, J.: Advances in underwater robot vehicles for deep ocean exploration: Navigation, control, and survey operations. In: Robotics Research - The Ninth International Symposium. (1999)
7. Šišlák, D., Rollo, M., Pěchouček, M.: A-globe: Agent platform with inaccessibility and mobility support. In Klusch, M., Ossowski, S., Kashyap, V., Unland, R., eds.: Cooperative Information Agents VIII. Number 3191 in LNAI, Springer-Verlag, Heidelberg (2004)
8. A-globe: A-globe Agent Platform. `http://agents.felk.cvut.cz/aglobe` (2005)
9. The Player/Stage Project: Open Source development tools for robot and sensor applications. `http://playerstage.sourceforge.net` (2005)

A Reference-Model for Holonic Supply Chain Management

Richard Peters[1] and Hermann Többen[2]

[1] Mittenwalder Str.51, 10961 Berlin, Germany
skamtin@web.de
[2] Technische Universität Berlin,
Franklinstr. 28/29, 10587 Berlin, Germany
Hermann.Toebben@sysedv.cs.tu-berlin.de

Abstract. IT systems are getting more connected, more network-like and hence more complex. The question arises, how such systems with steadily increasing complexity could be managed? Therefore new innovative approaches like Arthur Koestler's Holon approach are needed. This method arose in the 60´s of the past century as a philosophical work based upon a collection of parables, but not much tested and used yet. This paper describes how this approach is deployed for the Supply Chain domain as a reference model for Supply Chain Networks. It can be shown, that based on better methods for the complexity management better efficiency as well as effectiveness can be achieved.

1 Introduction

In general systems are getting more connected, more network-like and even more complex. As one of those this paper presents a reference-model the supply chain network domain. Therefore it makes use of the philosophical thought about system theory from Arthur Koestler [12]. His idea of systems helps to manage the increasing complexity of those systems. The paper is structured as follows: first a short overview about Koestler's main idea and conception of a holon is given. Furthermore already existing approaches are addresses, which make also use of the holonic concept, and are compared with the here presented approach. It follows a brief description of the Supply Chain Networks and especially of the SCOR-model. Then the holon-based reference model is introduced. In the end the static and dynamic behaviour of this model are discussed and evaluated by scenario example concerning the well known bullwhip-effect.

2 The Holon Approach

Koestler has designed a system-theoretic model of Self-regulated Open Hierarchical Order (SOHO), which instead of mathematical symbolics is originally based upon parables. From Koestler's point of view there did not exist a sufficient mathematical theory for his thoughts at that time [11].

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 221 – 232, 2005.

Koestler realized that all complex structures and stable processes present a hierarchical structure, like living organisms, social societies and non-living systems. Complex systems evolve in much shorter time out of simpler systems, with stable character shapes between the complex and the simple ones [22]. Systems that start up simple and than turn from stable character shapes to complex systems are inevitably hierarchical systems [11].

Hierarchies. The establishment of a hierarchy is the base for structuring and the control within a system. Typically hierarchies have top-down-flow of orders (output-hierarchies) and a bottom-up-flow of information (input-hierarchies). In hierarchies the principle of feedback exists immanent – like in the control loop – where the outcomes from changes are compared to the goals. Most of the time hierarchies are looked at as rigid and inflexible shapes. However, some approaches are not following this like the approach for "Dynamical Hierarchies of Artificial Life" and the "Holonic Ansatz" according to Rasmussen [18].

From the hierarchical principle to the holonic approach. The evolutionary stability of complex systems build out of simpler subsystems in nature reflects their remarkable autonomy and independence. Every subunit could operate as a quasi-autonomic whole. They are sub-wholes facing their subordinated units as a whole, and in relation to their superordinated system as a dependent part [11]. This could also be an approach to integrate systems based on swarm algorithms which usually show emergent behaviour into a huger and more complex system. Even this whole-part dualism is important for Koestler. By definition 'part' is something fragmented and incomplete, that has no justification to exist for its own. To be the part of something complies the requirement for integration. The term 'whole' stands for something complete, which needs no further explanations. For a whole-parted system Koestler has invited the term 'holon', derived from the greek word holos = whole with the suffix –on (like neutron or proton) pointing out the part characteristic [11].

Tendency of autonomy and integration. A base for the stability of a holon - one of it's specific design pattern - are the internal rules or more precise a canon out of them. The rules describe the content, the structural configuration and the function-patterns. While the rules define possible actions, a strategy chooses the current action inline with the environmental requirements. A holon could compile a strategy out of it's rules, which fits to its intentions, goals and the interpretation of the environment.

The second feature of a holon is the tendency to integrate into a more comprising wholeness. The communication and cooperation skills of a holon emerge from this tendency to integration. Still not any complex is a holon. The complex must have a hierarchical order, rules and strategies. Without communication and reaction abilities it cannot be called a holon. Accumulations of artefacts are not holons, but they could be part of them.

Self-regulation. The principle of self-regulation is also fundamental for the concept of autonomy. If a holon is a semi-autonomic whole-part, then it must have plans for self-regulation. Put it briefly, an action must on one hand be conform to the inner rules of the holon and on the other hand depends on the observed environment variables. A permanent flow of information from the action – being executed – has to

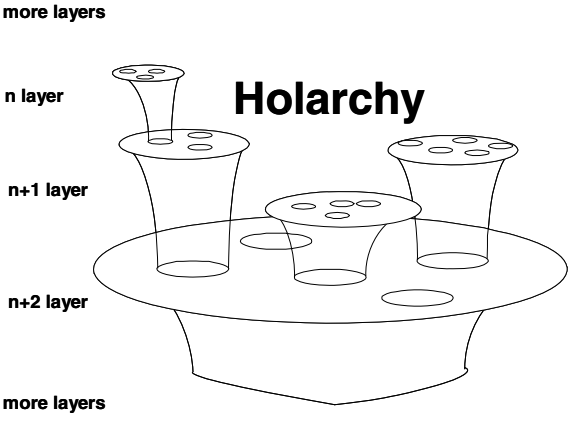


Fig. 1. Picture of a hierachical order of holons called holarchy [16]

exist to the part, which controls it. The actions have to be permanently regulated, which corresponds to the principle of feedback-control [12].

Holarchy. Hierarchies of complex systems containing not only simple connection structures, but also including heterarchic subsystems are called holarchies by Koestler. The latter are orders of holons, which have hierarchic dependences. Every holon could have an other internal structure (cmp. figure 1).

The higher layers of a holarchy are normally not in direct contact with the lower ones and vice versa. Signals pass established channels from one layer to the next, one step at a time, up or down in the hierarchy. A short circuit of the information flow above more layers could lead to unpredictable disturbances of the whole system (cmp. [12]).

With every step upwards in the holarchy, holons are getting more complex, flexible and unpredictable in their behaviour structures, but with every step downward the holons become more mechanical, stereotype and predictable in their behaviour structures.

Other holonic approaches. So far existing holonic applications or models apply mostly to the inner production processes of a company, called holonic manufacturing systems (HMS). Similar to all these approaches is that they are modelled domain specific. Mentioning a few of them: a holonic reference architecture for production systems, called PROSA, developed at the PMA-KULeuven by Wyns [24]. A second approach from Fischer [8] uses an agent platform to develop an HMS. Also HMS are build by Goua et al. [10], Adelberg [1] and Silva/Ramos [21]. Instead Bussmann [4] builds not a HMS but a holonic transportation management system. The here proposed reference model is related to all these approaches as a framework. Hence all the above mentioned systems but also legacy systems could be integrated into a holonic supply web, each of them as a concretisation of a specific holon, but mostly at the operative level of the model. You could also use different HMS at the same time in different holons.

3 Supply Chains

The holonic view served the handling of complex systems. According to Deloitte supply networks are complex systems and their complexity will even grow in the next time [7]. To explain the character of a supply chain, we give two example definitions:

A supply chain described from the macro-perspective corresponds to the stepwise transformation from raw material to end products and their distribution and selling to the customer. From the perspective of a single company (micro-perspective) it described the cooperation with its supplier and customer by the creation of value (cmp. Delfmann/Albers [6], p.42]).

A supply chain is a compound of activities, which contains planning-, coordination- and controlling-tasks. These activities are subordinated to the goal to produce the end product. Seuring points out, that the supply chain contains not only the physical motion of the good, but also the delivery management, the supply management, the production management, the material planning, the location planning, the customer service and the information flow [20].

For the building of the reference model it is resorted to quasi-standards in the supply chain domain. It is used the SCOR-model from the Supply Chain Council and an expansion of it, the task model from the SCM-CTC (Supply Chain Management–Competence & Transfer Center from the Fraunhofer Institut [13]) as starting points.

The SCOR model describes all enterprise activities, which are related with the satisfaction of the demand. With regard to figure 2 it subdivides into a four-layer structure, refining down to more detail layers. The first three-layers are of more

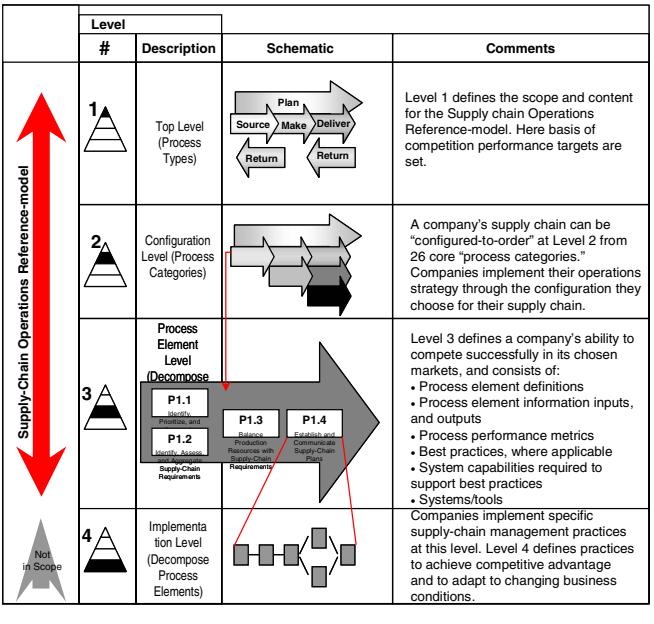


Fig. 2. Process layers of the SCOR model (Source: SCOR5.0 2001 [19], p.7])

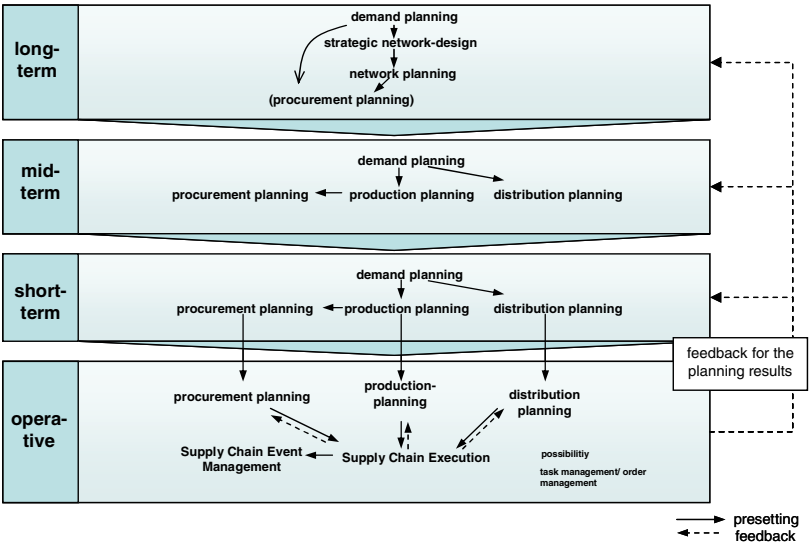


Fig. 3. Task model of the SCM-CTC (Source: Peters 2004 [17], p.31)

conceptual character, the lowest layer represents the implementation layer, which depends on the individual enterprise to enterprise and is not further explained in the SCOR model. The reference model refined here on a holon basis does not make use of all layers in the SCOR model, hence only specific functions are integrated. The task-model of the SCM-CTC can be seen in figure 3.

4 The Holonic SCM Reference-Model

The model separates into a static and a dynamic view. In figure 4 nearly every holon as designed for the reference model can be seen. In opposition to the static view holons could also change within time, e.g. the distribution holon could be coupled out and become an enterprise holon on its own. This is the reasons, why dynamic behaviour is of that special interested. Before specifying this, a short overview above the designed holon of the reference model should be provided. First all terminable holons had to be identified. Also subsequent expansions should be supported by them. Every holon has to fit into the holonic structure, meaning that every holon has to fulfil the grounding paradigms of the holonic structure: every holon has to be able to communicate (the integration-paradigm) and acts by its own driven by rules and strategy (the autonomy-paradigm).

The enterprise holon represents location point of a firm as a whole. The location is simply a node of the supply network. The enterprise holon consists out of following subholons: the sourcing holon, the production holon, the distribution holon, a central service holon and finally a supply net planning holon. The first three of them are also called process holons and have been derived from the SCOR model. This holon building block set should be sufficient to model every thinkable company structure.

The major task of the enterprise holon is to coordinate it's subholons respectively to give them the platform for their own self-coordination. In favour of the task fulfilment the enterprise holon defines the directive structure of subholons from the next lower holarchy level. An additional task of the enterprise holon is to conduct changes: the enterprise holon stimulates for capacity expansion and reduction. If necessary it could replace subholons and give the impulse for the change of the internal structure of a subholon. At the outer interface the enterprise holon communicates with other enterprise holons to optimize the supply chain network.

The supply net planning holon comprises for all skills necessary for the long-term planning. The process holons are responsible for the mid- and short-term planning. The operative holons of the process holons and the security-/monitor holons take over the operative planning tasks. The supply net planning holon is responsible for the integration of a company into the overall planning of the supply net, which complies to the tasks of the strategic network design, cooperative demand planning and the network planning. The planning task is solved in cooperation with the other network nodes. Results are then forwarded to the process holons.

The sourcing holon is namely responsible for all aspects of sourcing. It consists of a planning holon, an operative holon and a security/monitor holon. The planning holon takes all planning tasks, the operative holon takes all process task while the security/monitor holon supervises the planned processes. The production holon is responsible for all aspects of production and the distribution holon for all aspects of distribution. Their internal build-up is same like the sourcing holon.

The central service holon provide services that the process holons will needed, like bill of cost, internal transportation and storage.

Only the dynamic behaviour of supply chain networks could let you have a presentiment of the true complexity of such systems. You could compare it to circular flow of a higher organism. The problem is not to design any complex system, but to design it such that it shows up emergent behaviour. Bonabeau exemplifies this [3]:

"… using a swarm-intelligent system to solve problem requires a thorough knowledge not only of what individual behaviour must be implemented but also what interactions are needed to produce such or such global behaviour."

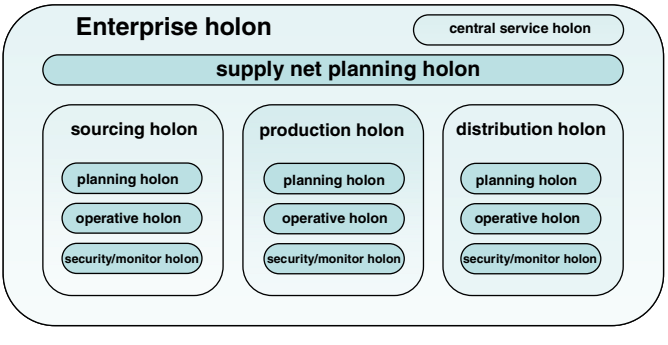


Fig. 4. Static view on an enterprise holon (Source: Peters 2004 [17], p.47)

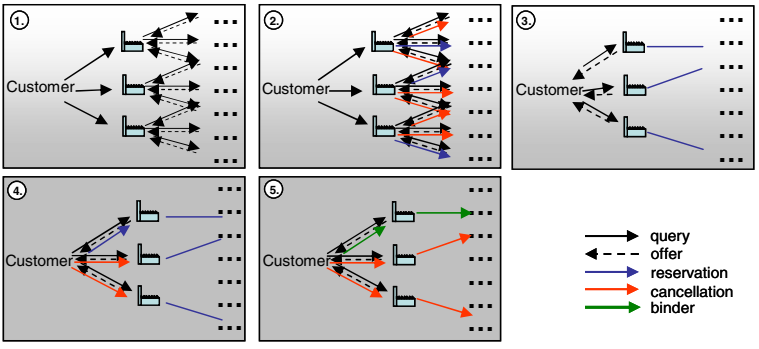


Fig. 5. Short-time negotiations (Source: Peters 2004 [17], p.64)

In the holonic reference model the supply chain network nodes interact with each other on different layers. The activities of the supply network have different structures. Some of the information flows are pyramidal from the customer to supplier and back, while others pass through all parts of the net. Accordingly the complexity of supply systems the dynamical behaviour for negotiations, the material flow including the collocated information flow, the order management, the control- and monitor-functions and finally the cooperative planning have been designed (cmp. figure 5).

The negotiations could be divided into long-term and short-term ones. The results of the long-term negotiations are of more timely stability. The procedure for the short-term negotiations can be seen in figure 5 and explained as enlisted in the following:

- Point 1: The customers query
- Point 2: The first nodes ask their suppliers
- Point 3: Answer
- Point 4: Reservation of the best offer, the others get refusals
- Point 5: Offer to the customer
- Point 6: The best offer is chosen, the others get refusals
- Point 7: The reservation of the best alternative is transformed into a concrete order, the reservations of other vendors get refusals

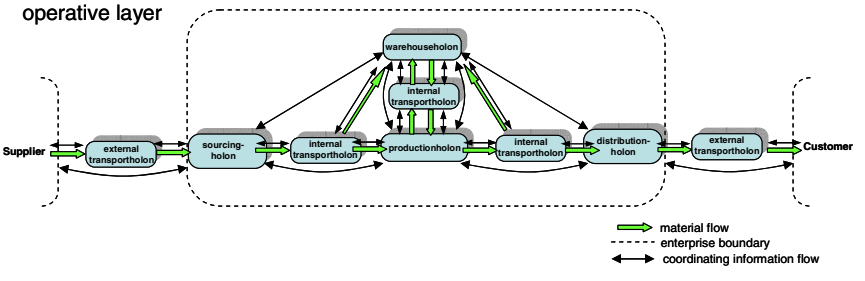


Fig. 6. Drawing of the material flow including the collateral information flow (Source: Peters 2004 [17], p.75)

The material flow as depicted in figure 6 is attended with an agile information-exchange, like mail conformations, planning data, error messages and so on. The tasks that occur during the material flow are taken over by the operative holons, like the transformation, the transportation and the storage of the goods.

Order management. The order holons are the lowest entities in the holarchy that is designed in this reference model. They are placed within the operative holons of the process holons. Every order gets an order holon, so the order holons forms order pyramids depended on the first order or the end-customer order. In a network so many order pyramids exist like end-customer orders exist. So this order pyramids could interact with each other or alone to coordinate the order fulfilment. If an order gets late, the depended orders could higher their negotiation might for the resources and try to catch up the lost time. This could be an adequate mean against the ripple-effect.

"…the variation of leadtimes at any stage will affect the execution of the other stages and result in uncertainties for the overall order cycle time. This is called the ripple effect." – Lin et al. 2000 [15], p.234 –

The monitoring of the material flow is one part take over by the order holons in a permanent self-control and the other part by the security/monitor holons. The security/monitor holon is responsible for identification and forwarding of error messages from one process holon to the next. It activates new planning (reactive planning) and tries to embank the error causalities. The philosophy is to embark the errors as early as possible, so that many of the supply chain partners will hold their planning security.

In figure 7 the coherence of the different planning holarchies could be seen, like the overall coordination in the network, the coordination in view of an order holarchy, enterprise holon internal directive structure, feedback-loops. The overall network coordination is in the simplest case the forwarding of prognosis, at the time data and enterprise data, but it can also show swarm behaviour, so that the neighbours give input to expand or reduce capacities.

Building a system with central or decentral coordination based on this reference model is a modelling task. This reference model should be seen as a construction set for complex systems that support both central as well as decentral solutions. It is

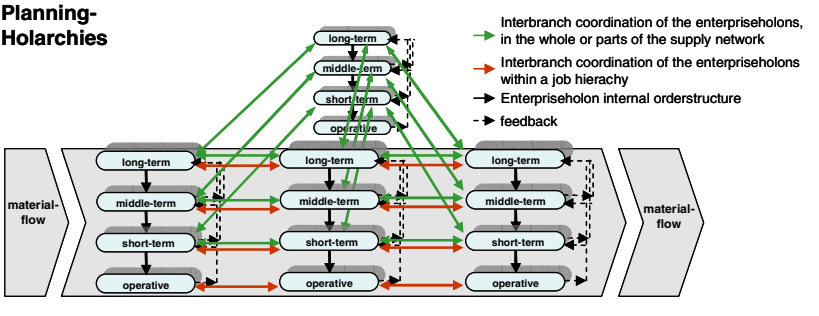


Fig. 7. Cooperative planning (Source: Peters 2004 [17], p.89)

possible to model one layer of the holarchy centrally but another one decentral, if needed. Just since many supply chain systems use agent technology for implementation, according to Dangelmeier agent technology could also be used for holonic applications [5]. This technology will support your design decision no matter if central or decentral. For more information about agent technology is referred here to Glückselig [9] and Peters [16].

5 Evaluation for the Bullwhip-Effect

Since the here proposed reference model has not been practically implemented, its usefulness has been evaluated by executing scenarios of practical relevance. One of this regards the well-known bullwhip-effect and is exploited in more detail. If significant fluctuations occur within a supply chain, but the end-consumer demand is relatively constant, this phenomenon is called the "bullwhip-effect". According to Delfman/Albers [6] the reason for this is, that the actualisation of sales prognoses, packaging of orders, price fluctuation and rationing-and-shortage gaming lead to incremental amplification. Lee pronounced that each of the four forces in conjunction with the chain's infrastructure and the order managers' rational decision making create the bullwhip effect [14].

Table 1. Bullwhip-effect (Source: Störk 2003 [23], p.4)

Time	0	1	2	3	4	5	6
customer-demand	100	100	103	100	100	100	100
dealer-inventory stock	100	100	97	103	103	100	100
dealer-calculated assets		200	203	203	200	200	200
dealer-order size	100	100	106	100	97	100	100
wholesaler-inventory stock	100	100	94	106	109	97	97
wholesaler-calculated assets		200	206	206	197	197	200
wholesaler-order size	100	100	112	100	88	100	103
distributor-inventory stock	100	100	88	112	124	88	85
distributor-calculated stock		200	212	212	188	188	203
distributor-order size	100	100	124	100	64	100	118
producer-inventory stock	100	100	76	124	160	64	46
producer-calculated assets		200	224	224	164	164	218
producer-order size	100	100	148	100	4	100	172

The example chosen from Störk [23] concentrates on consequences caused by package of orders along a supply chain. In figure 8(Left) and table 1 increase of the inventory by the disturbance of 3% of the demand can be observed. In figure 8(Right) and table 2 the changed demand will be passed collaboratively along the supply chain partners and the bullwhip-effect does not occur. The example is simplified, but shows the difference between collaborative planning and not coordinated inventory policies. The following equations are the basement for the bullwhip-effect (see table 1):

$$\text{demand} = \text{order size of the following step} \tag{1}$$

inventory stock(t) = inventory stock (t-1) – demand (t) + order size (t-1) (2)

calculated assets (t) = demand (t) + demand (t-1) (3)

order size (t) = calculated assets (t) – inventory stock (t) (4)

In table 2 it can be observed, how simple coordination of the information flow could stop the bullwhip-effect in this example. The difference between table 1 and table 2 is that the information of the customer demand is passed through the supply chain, so every company could calculate the overall demand:

demand = for every node the customer demand (5)

inventory stock(t) = inventory stock (t-1) – demand (t) + order size (t-1) (6)

calculated assets (t) = demand (t) + demand (t-1) (7)

order size (t) = calculated assets (t) – inventory stock (t) (8)

Table 2. No bullwhip-effect, (Source: Peters 2004 [17], p.97)

Time	0	1	2	3	4	5	6
customer-demand	100	100	103	100	100	100	100
dealer-inventory stock	100	100	97	100	100	100	100
dealer-calculated assets		200	203	203	200	200	200
dealer-order size	100	100	103	100	100	100	100
wholesaler-inventory stock	100	100	97	100	100	100	100
wholesaler-calculated assets		200	203	203	200	200	200
wholesaler-order size	100	100	103	100	100	100	100
distributor-inventory stock	100	100	97	100	100	100	100
distributor-calculated stock		200	203	203	1200	200	200
distributor-order size	100	100	103	100	100	100	100
producer-inventory stock	100	100	97	100	100	100	100
producer-calculated assets		200	203	203	200	200	200
producer-order size	100	100	103	100	100	100	100

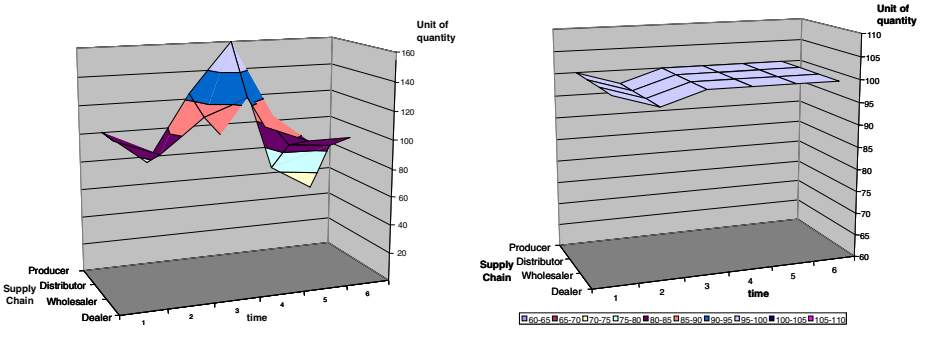


Fig. 8. Left: Inventory with bullwhip effect(Source: Peters 2004 [17], p.93) Right: No bullwhip-effect via collaborative planning (Source: Peters 2004 [17], p.98)

6 Summary and Outlook

The reference model for the supply chain management as presented here has been designed using Arthur Koestler's holonic approach. In favour of a more throroughly analysis especially in cases of great system complexity, the next step is to develop a prototype, where such systems could be simulated and strategies be developed. Especially the de-escalation problem of increasing complexity in the management of supply networks using holon-structuring is one of the great advantages of this approach. In order to manage a complex system which contains many different systems and hence several sub- and sub-sub-systems, automatically leads to the phenomena of emerging systems. The domain of supply chain networks is a excellent research field for this phenomena. A transfer of the results and strategies from this research field into practical domains is the major goal following Bertalanffy's general system theory [2].

References

1. Adelberger, H.: Economic Coordination Mechanisms for Holonic Multi Agent Systems. In: 11th International Workshop on Database and Expert Systems Applications (DEXA'00), p. 236, Greenwich, London 2000 at the 3.1.2003: http://nestroy.wi-inf.uni--essen.de/~conen/conen/paper/holomas-00/
2. Bertalanffy, L.v.: General System Theory. 14'th printing, USA 2003
3. Bonabeau, E., Dorigo, M., Théraulaz, G.: Swarm intelligence: from natural to artificial systems. Oxford 1999
4. Bussmann, S., McFarlane, D.C.: Rationales for Holonic Manufacturing Control. In: Proc. of 2'nd Int. Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, 1999, pp. 177-184
5. Dangelmeier, W., Pape, U., Rüther, M.: Agentensysteme für das Supply Chain Management – Grundlagen – Konzepte – Anwendungen. Wiesbaden 2004
6. Delfmann, W., Albers, S.: Supply Chain Management in the Global Contex. Köln 2000. Access: 2.7.2004: http://www.uni-koeln.de/wiso-fak/planung
7. Deloitte Toache Tohmatsu: The challenge of complexity in global manufacturing – critical trends in supply chain management. Access 5.6.2003: http://www.deloitte.com/dtt/cda/-doc/content/SupplyChainSurvey%2813%29.pdf
8. Fischer, K.: Agent-Based Design of Holonic Manufacturing Systems. In: Journal of Robotics and Autonomous Systems, Elsevier Science B.V., 1999. Access http://www.dfki.de/~kuf/papers/BASYS-HMS-Journal.pdf
9. Glückselig, S.: Holonische Multiagentensimulation. Diploma Thesis, Würzburg 2005
10. Goua L., Luhb, P. B., Yuji Y.: Holonic Manufacturing Scheduling: Architecture, Cooperative Mechanismen and Implementation. In: Proceedings of SPIE, Vol. 3203, 1997
11. Koestler, A.: Jenseits von Atomismus und Holismus – der Begriff des Holons. In: (Hrgs.) Koestler, A., (Hrgs.) Smythies, J.R.: „Das neue Menschenbild – Die Revolutionierung der Wissenschaft vom Leben", 1.Auflage, S.192-229, Wien 1970
12. Koestler, A.: The Ghost in the machine. 1. amerikanische Auflage, New York 1976
13. Lackmann, F., Nayabi, K., Hieber, R.: Marktstudie 2003 – Supply Chain Management Software – Planungssysteme im Überblick. Germany 2003

14. Lee,H.L., Padmanabhan, V., Whang, S.: The Bullwhip Effect In Supply Chains. Sloan Management Review, Spring 1997, Volume 38, Issue 3, pp. 93-102
15. Lin, F., Strader, T.J., Shaw, M.J.: Using swarm for simulating the order fulfilment process in divergent assembly supply chains. In: Luna, F., Steffensson, S. (eds.): Economic Simulation in Swarm: Agent-based Modelling and Object-Oriented Programming". Boston 2000, pp.225-249
16. Peters, R.: Entwicklung einer holonischen Modellierungsumgebung für Multi-Agenten Systeme. Studienarbeit, Technische Universität Berlin 2003. Access 28.1.2005: http://holon.gungfu.de/wiki/field.php?pagename=Main.RichardPeters
17. Peters, R: Entwurf eines Referenzmodells für Supply Netzwerke unter Verwendung des Holonen Ansatzes. Diploma Thesis, Technische Universität Berlin 2004. Access 28.1.2005: http://holon.gungfu.de/wiki/field.php?pagename=Main.RichardPeters
18. Rasmussen, S., Baas, N.A., Mayer, B., Nilsson, M., Olesen, M.W.: Ansatz for Dynamical Hierarchies. In Artifical Life. 2001; 7(4) page 329-353. Access 10.10.2004: http://www.mitpress.mit.edu/catalog/item/default.asp?tid=7678&ttype=6
19. Supply-Chain Council: Supply-Chain Operations Reference-model – SCOR 5.0. Supply-Chain Council, 2001. Access 2.7.2004: http://www.supply-chain.org
20. Seuring, S.: Die Produkt-Kooperations-Matrix im Supply Chain Management – Konzeption und instrumentelle Ausgestaltung. EcoMTex-Diskussionspapier Nr. 02, Oldenburg 2001. Access 2.7.2004: http://www.uni-oldenburg.de/produktion/download/-02_Seuring_SCM_Matrix.pdf
21. Silva, N.; Ramos. C.: Holonic Dynamic Scheduling Architecture And Services. In: International Conference on Enterprise Information Systems - ICEIS'99; Setúbal, Portugal, 28-30 March 1999
22. Simon, H.A.: Die Architektur des Komplexen. In: Die Wissenschaft vom Künstlichen. Wien 1994, 144–176
23. Störk, J.: 10. Planungssysteme. Folienvortrag, Oldenburg 2003. Access 15.6.2004: http://www-is.informatik.uni-oldenburg.de/~stoerk/lehre/planen03/Planungs-systeme_10.pdf
24. Wyns, J. : Reference architecture for holonic manufacturing systems – the key to support evolution and reconfiguration, (Leuven). Belgium 1999. Access 2.5.2005: http://www.mech.kuleuven.ac.be/~jwyns/phd/phd_jo_wyns.pdf

Polymorphic Agent Clusters – The Concept to Design Multi-agent Environments Supporting Business Activities

Waldemar Wieczerzycki

The Poznan University of Economics
w.wieczerzycki@ae.poznan.pl

Abstract. There are two main contributions of the work reported in the paper. First, a new model of software agents is proposed that are polymorphic and highly mobile. The former feature of agents results from the existence of potentially many agent versions. The latter feature is implied by the fact that only the necessary agent code is transmitted through the network, while the rest is transmitted only on demand.. Second, in the paper a particular approach to develop and manage multi-agent environments is proposed. It is based on so called agent clusters which group agents that mutually cooperate to perform a particular mission, e.g. to build and manage supply chains, to negotiate details of orders, to sign business contract. The approach seems to be straightforward, on one hand, and allows practically unrestricted collaboration among agents of the same business party, and safe cooperation among different business parties, on the other hand. The concept of agent cluster is inspired to some extent by the database technology, in particular by database transaction management.

1 Introduction

Agent technology is very promising for e-supply chain configuration and management [2] based on e-markets, what is strictly related to three particular properties of software agents. First, agents are autonomous, thus a user can initialize them (using a relevant application) and disconnect from the network, provided agent mission is well defined. For the purpose of agent initialization a wireless communication technology can be used, e.g. GSM. After some time the user can re-connect to the network checking if the mission has been already entirely fulfilled or not, and get final or partial results of agent work. Second, agent can be highly mobile what is very advantageous and required in supply chain configuration and management. Agents accessing e-markets can lead to dynamic supply chain reconfiguration, what can happen very often, even every week – this opens previously unknown opportunities for enterprises which can avoid stable supply chains, built on the basis of long-term contracts with rarely replaced business partners. Third, agents can be very intelligent, thus they can support efficient supply chain configuration including partners mutually offering at a given time moment the best cooperation possibilities and conditions.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 233 – 245, 2005.

In case of supply chain configuration [4] there is an evident need to browse the offers available on a large number of e-markets. Similarly, in case of supply chain management, there is typically a need to visit many nodes of chain, e.g. following product or service flow over the chain. Thus, logistics requires agents which are both: highly mobile and very intelligent, that could efficiently adapt to diverse environments of business partners, their specificity and results expected.

Agents intelligence is extremely required if they are assumed to perform cooperative activities in multi-agent environments visited. Notice, that in case of e-markets, an agent typically not only browses available offers. If some of them are potentially interesting then the agent, in the name of the delegating user, can start collaboration either with the agents representing producers of the considered offers (i.e. parties announcing their supply) or with the producers' representatives themselves [6]. In the latter case the agent can communicate with the representatives (i.e. users) through local applications executed in parallel with the multi-agent environment. In both cases the agents cooperative activities consist mainly in negotiations, e.g. of prices, delivery conditions, amount of products, products quality. If the negotiation is successful then the agent can even digitally sign business contract.

As it is well known, the aforementioned agents properties (namely mobility and intelligence) are in practice in opposition, i.e. very intelligent agents have low mobility (due to their cost of transmission time) and vice versa. Thus the application of classic agent technology in logistics is not recommended, since it leads to "fat" and slow agents. On one hand, their code is long because they must be multiversion - in order to be universal enough to easily adapt to versatile environments. On the other hand, they continuously grow when moving over the network nodes, as result of knowledge (data) acquisition.

In such situation the first goal of the paper is to propose a new model of software agents that are polymorphic (and more intelligent as a consequence) and highly mobile. Speaking very intuitively, the first feature of agents results from the existence of potentially many agent versions. The second feature is implied by the fact that only the necessary agent code is transmitted through the network, while the rest is transmitted only on demand. Moreover, the data which does not extend agent intelligence, but only its knowledge, is not carried between nodes by the agent, but sent directly to a sort of data repository which is stationary.

Every electronic market (e-market) is based on a centralized or distributed information repository. This repository contains a variety of multimedia business documents, e.g. product and service offers, product and service needs, business contracts, negotiation reports and protocols. In the simplest case the information repository is stored in a file system. However, in more advanced e-market applications the repository is stored and managed by a database management system (DBMS).

Business processes of e-markets have transactional nature. To some extent they should have atomicity, consistency and persistency properties. They also should be supervised in order to detect and resolve access conflicts concerning pieces of business information stored in the repository. As a consequence, in case of e-market based on a file system, there is an evident need for a multi-agent environment that provides contexts for execution of e-market processes. In practice it means that the environ-

ment should provide some mechanisms used in database technology, properly adapted to the specificity of e-market functionality, on the one hand, and agents behavior, on the other hand.

In case of e-market based on a database management system there is also a need for an agents environment, since classical database transaction mechanism is too restrictive for e-markets, and excludes collaboration between e-market real users or software users (agents) [3]. Thus, in this case, the environment must somehow relax constraints implied by the DBMS being a kernel of e-market application.

In both cases the agents environment should be also responsible, on one hand, for a proper support of information sharing and exchange between agents, what is typical for e-markets. On the other hand, the environment has to substantially support structured multi-attribute negotiation among agents which is crucial in all phases of business transactions [1],[5]. Speaking more generally, the environment has to support safe, reliable and efficient cooperation among e-market participants, possibly with no limitations.

Thus the second goal of this paper is to propose a specific environment for polymorphic agents that can be useful in the development of e-market applications that fulfill the requirements mentioned above.

The structure of the paper is as follows.

Section 2 presents the PSA agent model which is based on multi-dimensional versioning and segmentation of agents code, as well as on distribution of agents code and data over the network. The role of proxy agent and bootstrap agent is detailed. Also the rules of agent migration, intelligence enrichment, adaptation to environments visited and self-slimming are discussed. The architecture of the proposed polymorphic agents is related to the FIPA specification (Architecture Specification), which defines agents environments and relations between their elements [10].

In Section 3 multi-agent environment supporting collaboration of agents built according to the PSA model is presented. First, basic concepts of the approach are given. Next management problems concerning agents collaboration are described. Finally the discussion is focused on negotiation of access conflicts. As mentioned before, the approach proposed in this section is inspired by selected mechanisms of the database technology which plays the role of a core technology in the development of business information systems.

Section 4 concludes the paper. In particular it emphasizes additional advantages resulting from building e-markets based on the PSA agents model and agent clusters. It also briefly comments related work concerning programming language extensions towards modeling agents according to the PSA model, in particular their multi-dimensional nature and behavior.

2 The PSA Agent Model

According to the PSA (Polymorphic Self-Slimming Agent) model, an agent is composed of a sort of agent head, called *bootstrap agent*, and agent body. Bootstrap agent is relatively small, thus it can be highly mobile. Its goal is to move over the network and to decide whether a newly visited environment is potentially interested taking into account agent mission defined by the user. If it is, then the bootstrap agent recognizes

the specificity of the environment, afterwards it communicates with so called *proxy agent*, residing on the origin computer (i.e. a computer in which agent has been created by the user), asking for sending to the bootstrap agent an appropriate part of the code of agent body. Bootstrap agent communicates directly with the proxy agent using the messages in an agent-communication-language (ACL) [9], [11].

It may happen that after some time, if the results of agent (i.e. bootstrap agent extended by agent body) activity are satisfactory, the bootstrap agent again communicates the proxy agent asking for the next part of agent body. This situation will be explained later.

If agent mission in the currently visited environment is completed then the agent body is removed from the environment and the bootstrap agent migrates to a new environment or it returns to the origin computer in order to merge with the proxy agent.

For the sake of platform independence and security, we assume that the bootstrap agent is interpreted by a visited environment. In other words the bootstrap agent is a source code, rather than binary code.

There are four basic functions provided by the stationary proxy agent:

- It serves as a communication channel between the user and bootstrap agent: it knows current location of bootstrap agent and can influence the path of its movement, as well as tasks performed by the bootstrap agent.
- It encompasses all variants of the agent's code that model the variability of agent behavior. Depending on what environment is currently visited by the bootstrap agent, and according to bootstrap agent demands, proxy agent sends a relevant variant of agent code directly to the bootstrap agent, thus enriching it with the skills required in this new environment and, as a consequence, enabling it to continue the global agent mission. Notice, that code transmission redundancy is avoided, since the unnecessary code (not relevant agent variants) remains together with the stationary component.
- It assembles data items that are sent to it directly from the bootstrap agent, extended by a proper agent variant, which are not useful for a mobile code, while could be interesting to the user. The data assembled is stored in so called *knowledge repository*.
- Whenever required, the proxy agent responds to the user who can ask about mission results and data already collected (e.g. a percentage of data initially required). If the user is satisfied with the amount of data already available, proxy agent presents the data to the user and finishes its execution (together with a mobile component).

Now we focus on possibilities of the PSA agent versioning, i.e. on the content of the proxy agent that is always ready to select a relevant piece of agent code according to a demand of the bootstrap agent. We distinguish three orthogonal dimensions of agent versioning:

- agent segmentation,
- environmental versioning,
- platform versioning.

Agent segmentation. Typically agent mission can be achieved by performing a sequence of relatively autonomous tasks (or stages). Thus the agent code is divided into so called segments corresponding to consecutive tasks that have to be realized. If one task is finished successfully, then the next one can be initiated. Thus, next segment of agent code is received from the proxy agent and agent execution switches to this new segment. Depending on whether agent behavior is sequential or iterative, the previous segment is automatically deleted (in the former case) or it remains in the execution environment (in the latter case).

For example, first segment of the agent can be used for browsing offers available at e-market. If there is an interesting offer, then second segment is transmitted on demand which is responsible for negotiation of terms of cooperation with the agent presenting the offer. And again, if the negotiation succeeds, then third segment is transmitted which is responsible for business contract signing, and so on. Contrarily, if a particular agent task fails, there is no need for next segment transmitting.

In the above example a sequential behavior of the agent would be recommended, in order to reduce segment transmitting. It means that the first segment should browse all offers available at the e-market, mark potentially interesting offers (i.e. worth to negotiate), and then switch to the second segment. Similarly the negotiation should be performed with all agents related to the marked offers. As before, if negotiation succeeds, second agent segment marks the respective agents, what is necessary for execution of the third segment.

To summarize, this versioning dimension, namely segmentation, models multi-stage nature of PSA agents.

Environmental versioning. Bootstrap agent can visit different environments providing different services, e.g. e-market, auction service, virtual aggregator of orders. Moreover, every service can be implemented in a different way. For example, the e-market can be implemented as a web-site, database or mailing list. Thus, depending on the specificity of environment being visited different version of agent segment is required.

In other words, the proxy agent keeps and manages sets of versions for each agent segment and takes care on their consistency, i.e. knows which versions can "go together". For example, if the environment currently visited is the e-market implemented as a web site, then versions of consecutive segments delivered to the visited environment must be consistent, i.e. they must match the specificity of e-markets being web-sites.

To summarize, this versioning dimension, namely environmental, models polymorphic nature of PSA agents.

Platform versioning. Finally, every version of every agent segment is available in potentially many variants which are implemented for a particular target environment (i.e. for a particular hardware which runs agent environment: processor, operating system, network communication protocols etc.). There is one particular variant for each agent segment version, which is in a source form. It is sent to the environments which for the security reasons do not accept binary (executable) code, it means which interpret agents instead running compiled code. Besides this particular variant, the proxy agent keeps potentially unlimited number of binary variants. If the environment

accepts binary code than the proxy agent delivers a variant matching hardware and system software parameters of this environment.

Of course, binary variants ensure efficiency of the agent execution, while a single source variant guarantees security of agent interpretation.

To summarize, this versioning dimension, namely platform, models platform independent nature of PSA agents.

Now let us illustrate the behavior of a PSA agent by the example which shows step-by-step typical scenario of migration of code and data over the network.

1. The user creates a PSA agent on his/her computer using a relevant application, defines its mission, environments to be visited, assigns his/her certificate, and finally disconnects or shuts down the application.
2. PSA agent starts its execution on the origin computer. It is composed of the bootstrap agent, the proxy agent and the knowledge repository.
3. The bootstrap agent migrates through the network to the first environment under investigation.
4. The bootstrap agent is interpreted by the visited environment; it checks its specificity and platform details; assume that it is e-market implemented as a web-site, running at the computer equipped with Intel Centrino Processor managed by MS-Windows XP operating system; the environment allows for binary agents execution.
5. The bootstrap agent contacts the proxy agent through the network asking for sending first agent segment, in a version relevant to the e-market being visited, and in a variant matching the hardware and software parameters already recognized.
6. First segment of the agent migrates to the environment visited (more precisely: a particular variant of a particular version of first segment migrates) and starts its execution – let's say offer browsing. The data collected by the segment which are not required in further execution, e.g. offers that could be interesting in the future, are sent back directly to the knowledge repository, thus self-slimming the agent. All offers that should be negotiated are marked by the agent segment which finishes its execution (its code is deleted).
7. The bootstrap agent contacts the proxy agent again asking for sending second agent segment, say responsible for negotiation.
8. Second segment arrives and starts its execution. For the sake of simplicity we assume that negotiation fails with all respective agents related to the offers marked by the previous segment.
9. The bootstrap agent is informed about negotiation results and the second segment is deleted.
10. The bootstrap agent migrates through the network to the second environment under investigation. Steps 4 – 10 are repeated in this environment.

3 Clusters of Collaborating PSA Agents

Assuming agents built and working according to the PSA model presented in the previous section, now we discuss a new mechanisms of multi-agent environment which support and facilitate cooperation among agents visiting the same environment.

The main idea of the proposed multi-agent environment design is the use of so called agent clusters. Before we introduce agent clusters, we have to define some basic concepts. Web servers built in agent technology (e.g. e-markets) can be accessed in practice by an arbitrary number of agents (e.g. representing many businessmen) which work independently, or collaborate with other agents, e.g. to dynamically re-configure electronic supply chain. Depending on whether agents collaborate or not, and how tight is their collaboration, we distinguish two levels of agents grouping: *constellations* and *clusters*. A constellation *Ci* groups agents which aim to achieve the common goal, typically to buy or sell products or services. Agents belonging to the same constellation can communicate with each other and be informed about progress of business efforts by the use of typical communication channels. Constellations are logically independent, i.e. an agent belonging to a single constellation is not influenced by the evolution of other constellations. It is possible for a single agent to contribute simultaneously in many constellations, thus the intersection between two constellations need not be empty. In this case, however, actions performed in the scope of one constellation are logically independent from actions performed in other constellations.

Agents belonging to the same constellation can collaborate tightly or loosely, depending on whether they represent the same business party, e.g. enterprise (possibly extended by business partners), corporation, or they represent different business parties, aiming to find new business opportunities. Tightly collaborating agents are grouped into the same agent cluster AC_i. Thus, a cluster is a agents subset of a corresponding constellation, with the restriction that a single agent A_i belongs in the scope of a single constellation exactly to one cluster, in particular, to a single-agent cluster. Of course, if the agent is included in many constellations, say n, then it belongs to n clusters.

A *semi-transaction* is a flat, ordered set of operations on the same business data repository (in particular - database operations) performed by agents of the same cluster, which is atomic, consistent and durable. In other words, a semi-transaction is the only unit of communication between a virtual agent representing component agents of a single cluster, and the business data management system (or DBMS).

Formally, a semi- transaction is defined as a triple:

$$ST = (Tid, Cid, ACid) \tag{1}$$

where *Tid* is a transaction identifier, *Cid* is an identifier of the encompassing constellation, and *ACid* is an identifier of the cluster to which *ST* is assigned.

Two semi-transactions from two different constellations behave in a classical way, which means that they work in mutual isolation, and they are serialized by data management system. In case of access conflicts, resulting from attempts to operate on the same data item in incompatible mode, one of transactions is suspended or aborted, depending on the concurrency control policy.

Two semi-transactions from the same constellation behave in a non-classical way, which means that the isolation property is partially relaxed for them. In case of access conflicts, so called *negotiation mechanism* is triggered by data management system (or DBMS), which informs agents assigned to both transactions about the conflict, giving them details concerning operations which have caused it. Then, using communication mechanisms provided by the environment, agents can consult their intended operations and negotiate on how to resolve their mutual problem. If commonly

agreed, they can undertake one of the actions described later in this section, in order to avoid access conflict, in particular they can negotiate mutual business needs and decide to become virtual business partners. If they succeed, transactions can be continued, otherwise classical mechanisms have to be applied.

A particular mechanism is used in case of operations of the same semi-transaction, if they are performed by different agents, and they are conflicting in a classical meaning. There is no isolation between operations of different agents, however in this situation so called *notification mechanism* is triggered by data management system, which aims to keep the agents assigned to the same semi-transaction (cluster) aware of operations done by other agents. We have to stress that it concerns only the situation when an agent accesses data previously accessed by other agents, and the modes of those two accesses are incompatible in a classical meaning. After notification, agents assigned to the same semi-transaction continue their work, as if nothing happened. Notice, that in case of agents of the same cluster, we assume not only strict collaboration, but also deep mutual "confidence".

Now we focus on operations which can be performed on semi-transactions being contexts of agent clusters activity.

Every semi-transaction is started implicitly by *initialize*(T_i) operation, which is performed automatically by the agent environment on the very beginning of respective cluster activity, i.e. after first data operation is requested by one of cluster members. This cluster member is called a *transaction leader*. *initialize*(T_i) is also triggered automatically, directly after one of cluster members performs explicit *commit*(T_i) operation, or implicit *auto-commit*(T_i) operation. All consecutive transactions of the same cluster are executed in a serial order.

After a semi-transaction is initialized by the transaction leader, other cluster members can enter it at any moment of the transaction execution, by the use of explicit *connect*(T_i) operation, which is performed in asynchronous manner. Once connected to the transaction, any member of a cluster can perform *disconnect*(T_i) operation, providing there is still at least one agent assigned to this transaction. *disconnect*(T_i) operation breaks the link between transaction T_i and the agent, which can next:

1. close its session,
2. suspend its operations for a particular time moment and re-connect to the same cluster later,
3. wait until transaction commits and connect to a next semi-transaction of the same cluster,
4. continue to work in different cluster in the scope of another semi-transaction, provideed the agent belongs to more than one cluster.

In cases: 1, 2 and 4, *disconnect*(T_i) operation plays the role of *sub-commit* operation, which means that the respective agent intends to commit its own operations, and leaves the final decision whether to commit or not the semi-transaction to its recent collaborators (agents).

Operations introduced up till now concern a single semi-transaction. Next two operations: *merge*(T_i) and *split()* are special, since they concern two semi-transactions. They are extremely important for supporting typical business activities (e.g. on e-markets). *Semi-transaction* T_j can merge into transaction T_i by the use of *merge*(T_i) operation, provided the members of a cluster assigned to T_i allow for it. After this

operation, transaction T_j is logically removed from the agents environment, i.e. operation *abort*(T_j) is automatically triggered by the environment, and all T_j operations are logically re-done by transaction T_i. These actions are only logical, since in fact operations of T_j are just added to the list of T_i operations, and T_i continues its execution, however, the number of agents assigned to it is now increased. It means, that until the end of T_i execution, the agents cluster assigned previously to T_j is merged into the cluster assigned to T_i. Of course, *merge*(T_i) operation is only allowed in the scope of the same constellation. Obviously *merge*(T_i) can be useful when an access conflict between two clusters of the same constellation arises. But in terms of business activities this operation allows to couple representatives of business parties which decide to cooperate (e.g. negotiate cooperation conditions, browse proposals of business contracts and other documents, or even electronically sign them finally). Its use is more detailed further in this section.

Similarly to *merge* operation, *split()* operation can be used in order to avoid access conflicts. *split()* operation causes that a single semi-transaction T_i is split into two transactions: T_i and T_j. After *split()* operation, a subset of cluster members, originally assigned to T_i, is re-assigned to newly created transaction T_j. Also all operations performed by re-assigned agents are logically removed from transaction T_i and redone in transaction T_j directly after its creation. Notice, that in practice this operation can be useful if cooperation of representatives of two business parties for some reasons fails, and they decide for example to autonomously analyze the e-market to look for new parties offering them better cooperation conditions.

Contrarily to *merge* operation which is always feasible, provided members of the other cluster allow it, *split* operation can be done only in particular contexts. Speaking very briefly, a semi-transaction can be split if two sub-clusters, which intend to separate their further actions, have operated on disjoint subsets of data, before *split* operation is requested. If the intersection between the data accessed is not empty, *split* operation is still possible, provided the data have been accessed by the two sub-clusters in a compatible mode (in a classical meaning).

Finally, there are two typical operations on transactions: *commit* and *abort* which are performed in the classical manner (according to the database technology).

3.1 Semi-transaction Management

Let us remind, that the main assumption of the approach proposed in this section is to support collaboration among PSA software agents (cf. Section 2) representing business partners to possibly maximum extent, by providing them with mechanisms for:

- simultaneous work on the same pieces of information,
- mutual awareness of the current state and progress in cooperative business work, as well as notification about important operations done by members of the same agent cluster,
- negotiation, aiming at resolving potential access conflicts, resulting from incompatibility of operations requested.

The above mechanisms require additional information about intended operations and data locks implied by them to be collected by data management system. In our approach every lock (set or requested) is represented by the following quadruple:

$$L = (lt, C\_id, AC\_id, A\_id) \tag{2}$$

where *lt* denotes lock type, *C_id* and *AC_id* identify, respectively, the constellation and the agent cluster which encompass the transaction requesting a lock, while *A_id* identifies one of agents associated to semi-transaction.

Now we focus on notification and negotiation mechanisms. A notification call is sent from the agent environment to the application level. Its scope is limited to a single semi-transaction. When received, the application (e.g. auction system) informs the respective agent about the occurrence of a particular event, using available communication means.

When a negotiation mechanism is triggered, the following scenario is used:

1. the environment informs the agent holding a lock on a data item that another agent attempts to access this data in incompatible mode,
2. a data item concerned is indicated to the agent,
3. a communication link between two negotiating agents is established, by the use of available techniques,
4. the environment provides a solution which, if accepted, leads to a conflict avoidance,
5. agents negotiate, giving the details about intended operations on data item concerned,
6. if the solution available in the environment is commonly accepted, communication link is closed and negotiating agents continue their work; otherwise, the environment repeats all the steps again providing different solutions.

3.2 Possible Results of Negotiations

The following resolutions of access conflicts are proposed: fine-grained locking, commit request, lock release, operation embedding, transaction merge, transaction split, delayed versioning.

Fine-grained locking is a well known solution from database technology which is used in hierarchical locking methods. It consists in the conversion of a lock set on a big granule (e.g. a negotiated contract, a contract paragraph) into a set of locks set on smaller granules (e.g. paragraph clauses).

Commit request. This solution is quite straightforward. As it is well-known, in the classical two-phase locking approach locks are held until transaction commitment. During the negotiation, the holder of a lock is asked by the collaborating agent to commit its updates and start a new semi-transaction. If the agent agrees, a lock requested may be granted immediately after transaction commitment.

Lock release. There are two possible variants of this idea: operation roll-back and operation relegation. In the first variant, the negotiating agent (N1) which requests a lock asks the holder of a lock (N2) to roll-back operations done on the data item concerned. If accepted, the operation roll-back is followed by lock release, which makes it possible for both agents to continue their work. The rolled-back operation can be

redone by agent N1, or at least can influence the operations planned by agent N1 on a conflicting data item.

In the second variant, i.e. operation relegation, if the negotiating agent holding a lock (N2) agrees, then its operation on the data item concerned is logically removed from its semi-transaction, and relegated to the semi-transaction of the agent requesting a lock (N1). In other words, negotiating agent N2 rolls-back its operations on the data item concerned and releases its lock. Next, the rolled-back operation is automatically redone by the environment in the scope of a semi-transaction of N1. Both negotiators can then continue their work with the assumption that N1 is responsible for a proper state of a conflicting data.

Operation embedding. In this solution a negotiating agent which can not set a lock (N1) informs the holder of an incompatible lock (N2) about its intentions. If, after negotiations, N2 accepts the operation planned by its collaborator on the data item concerned, then this single operation is isolated from semi-transaction of the agent N1 and embedded in a semi-transaction of N2. In such situation, there is no need for lock release by N2, and for lock setting by N1, and both agents can continue their work. Notice, that this mechanism is based on a full confidence between negotiating agents. When N1 "convinces" its collaborator that its operation is reasonable, N2 allows the environment to embed a single operation of N1 into its semi-transaction.

Let us remind that in the above discussion, saying „a negotiator", we mean „virtual negotiator", which corresponds to an agent cluster, which may be composed of many tightly collaborating agents. Thus, in the negotiations participate in general more than two agents, but the final decision is undertaken by the transaction leaders. This remark is very important when we consider next two solutions.

Semi-transaction merge. Since one semi-transaction can not continue its execution because of lock set by another semi-transaction in the scope of the same constellation, then a natural solution is to merge those two transactions in a way presented before in this section. During the negotiation both agents: N1 and N2 consult their intended operations. If they agree that there is no conflict between their intentions, i.e. there is no conflict at the agents level, then the conflict at the environment level is superfluous, and the simplest way to avoid it is to merge both transactions, and continue a work in the scope of a single semi-transaction. The merge operation implies, of course, the merge of clusters.

Transaction split. This idea offers only a partial solution and can be useful if a semi-transaction ST which requests an incompatible lock is assigned to a cluster composed of more than one negotiating agent. In that case, the semi-transaction can split into two semi-transactions, in such way that the physical negotiator whose operation caused a request for lock setting is assigned to the first resulting semi-transaction (ST1), while the rest of physical negotiators are assigned to the second resulting semi-transaction (ST2). Now, semi-transaction ST2 can continue its execution, while ST1 has to wait until the requested data item is unlocked.

Presentation of the last resolution, called *delayed versioning*, would exceed the limit of the paper scope. It is detailed in: [7].

4 Conclusions

The following main advantages of the proposed approach can be distinguished:

- transmission of a minimum amount of code over the network (i.e. bootstrap agent and intelligence acquired as result of migration; agent variant code does not migrate with the bootstrap agent),
- reduced transmission of a collected data that is directly sent to the agent proxy and does not follow the mobile agent component,
- the possibility of monitoring the proxy agent by the user and obtaining partial data that can be satisfactory to the user,
- efficient support of collaboration with agents representing other business parties, being a consequence of potentially unlimited level o agent polymorphism.
- the semi-transaction model proposed for multi-agent environment is very straightforward and natural, on one hand, and allows practically unrestricted collaboration among agents of the same business party, on the other hand.
- potential conflicts between agents representing different parties are solved on a higher level than the level of a business information repository, as it happens classically; the environment presents to the agents available information on encountered management problems; then the agents can negotiate, presenting their intentions concerning future steps, and choose one of proposed mechanisms which aim at conflict avoidance.
- During cooperative work, the multi-agent environment supports agents awareness and notification, which are two very important functions of every business application.

References

1. Benyoucef M., Keller R.K., *A conceptual Architecture for a Combined Negotiation System*, 11th Workshop on Database and Expert System Management, 2000.
2. Denkena B., Zwick M., Woelk P.O., *Multiagent-Based process Planning and Scheduling in Context of Supply Chains*, 1st Int. Conference on Industrial Applications of Holonic and Multi-Agent Systems, Holomas 2003.
3. Elmagarmid A. (ed.), *Database Transaction Models*, Morgan Kaufmann, 1992.
4. Labarthe O., Tranvouez E., Ferrarini A., Espinasse B., Montreuil B., *A Heterogeneus Multi-agent Modelling for Distributed Simulation of Supply Chain*, 1st Int. Conference on Industrial Applications of Holonic and Multi-Agent Systems, Holomas 2003.
5. Rebstock M., *An Application Architecture for Supporting Interactive Bilateral Electronic Negotiations*, EC-Web Conference, 2001.
6. Ulieru M., *The Holonic Enterprise: Modelling Holarchies as Mass to Enable Global Collaboration*, 3rd Int. Workshop on Industrial Application of Holonic and Mult-agent Systems – HoloMAS, 2002.
7. Wieczerzycki W., *Advanced Versioning Mechanisms Supporting CSCW Environments*, Journal of Systems Architecture, Vol. 43, Issue 1-5, pp. 215-227, Elsevier Science.
8. Wieczerzycki W., *Software Reusability Through Versions*, Software - Practice and Experience, Vol. 26(8), pp. 911-927, John Wiley & Sons

9. www.fipa.org/specs/fipaSC00001L/, *Foundation for Intelligent Physical Agents*, FIPA Abstract Architecture Specification
10. www.fipa.org/specs/fipa00037/, *Foundation for Intelligent Physical Agents*, FIPA Communicative Act Library Specification
11. www.fipa.org/specs/fipa00061/, *Foundation for Intelligent Physical Agents*, FIPA ACL Message Structure Specification

Configuration of Dynamic SME Supply Chains Based on Ontologies

Eva Blomqvist[1], Tatiana Levashova[2], Annika Öhgren[1], Kurt Sandkuhl[1], Alexander Smirnov[2], and Vladimir Tarassov[1]

[1] School of Engineering at Jönköping University, Jönköping, Sweden
{eva.blomqvist, annika.ohgren, kurt.sandkuhl, vladimir.tarassov}@ing.hj.se
[2] St. Petersburg Institute for Informatics and Automation, St. Petersburg, Russia
{oleg, smir}@iias.spb.ru

Abstract. Due to the increasing implementation of agile and networked manufacturing, supply chain has entered a new phase, virtual supply chain. The phase is characterized by the integration of activities, operations, and functions carried out at different and geographically distributed supply chain stages. The paper proposes an approach to the configuration of a network of small and medium-sized enterprises (SMEs) being integrated into a supply chain. The SME supply chain configuration is based on a shared domain ontology for supply chain management, offering the configuration task as a function of supply chain management. Principles of the development of the shared ontology and possible ways of matching between enterprise and domain ontologies are considered.

1 Introduction

Networked organization structures have become common practice for small and medium-sized enterprises (SMEs) in order to strengthen their competitive positions. Examples of SME-networks include temporary project-based co-operations (e.g. in product design or system development projects), marketing organizations, and industrial clusters sharing expensive resources. SME-networks can be considered as virtual organizations (VOs) loosely integrating enterprises based on their contribution to the value chain [1]. VOs typically are governed by common economical and value-creation objectives and pro-actively form co-operations for a given demand. These co-operations are temporary, dynamical with respect to their members, geographically distributed, quick responsive and flexible to market demands.

The lifecycle of an SME-network includes six phases. The first phase, Community Building, results in a community of loosely coupled member enterprises with joint objectives and goals. This includes information exchange, coordination, and collaboration between the members. The Formation phase is initiated by a project. During this phase, definition of the project is modeled and partners are selected based on their capabilities. The third phase, Integration, results in a project network. It consists of negotiating business agreements, project execution modeling, and integration and con-

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 246 – 256, 2005.

figuration management. Operation is intended to carry out the collaboration project with relationships management and performance measurement. When the project network discontinues to exist, evaluation and disintegration on all levels with application of business agreements from Integration happens in the Discontinuation phase. The network is dissolved during the last phase, Community Redefinition.

Usually it is not fully obvious to the SME-network members, which competence and resources are available from which partner in which quantity to which expenses and how to access them. In this context, efficient support for configuration of co-operations and efficient reuse of existing knowledge is a critical success factor. Configuration includes selection of suitable partners based on their competences and integration of work processes and existing knowledge sources in order to ensure a common level of knowledge, understanding, and commitment. Ontological modeling of knowledge representation provides a way to achieve this common level.

We propose to use ontologies as a basis for SME supply chains configuration. Ontologies are widely considered as a technique for representing knowledge in an application domain. They have also been successfully applied in modelling competences of enterprises [2]. Our approach consists of four major elements: (1) capture all relevant characteristics of the overall application domain of a VO with domain and application ontologies; (2) represent the competences of VO members with enterprise ontologies; (3) identify candidates for SME networks based on matching between enterprise ontologies and domain ontology; (4) configure the SME supply chains by integrating enterprise ontologies of the identified candidates. This paper focuses on domain ontology related aspects, i.e. domain ontology development (contributes to element 1) and matching between enterprise ontology and domain ontology (contributes to element 3). Methodologies for enterprise ontology development and ontology integration are described in [3] and [4], respectively.

2 Application Ontology Development Methodology

The proposed methodology aims at the development of an ontology that could be used in problem solving. The methodology, like [5], focuses on the application-driven development of ontologies and shares views of Semantic Community Portals [6] in part of the creation of a rich domain ontology. Unlike [6], the developed domain ontology is to provide knowledge specific to a particular problem (task).

The conceptual framework is based on an analysis of research in ontology area [e.g., 7; 8; 9]. The methodology deals with the creation of ontologies of two types: an ontology describing knowledge of a certain domain (domain ontology) and an ontology describing what domain knowledge is required to solve a task (application ontology) (Fig. 1). These knowledge types lead to three levels the methodology addresses. The domain level describes domain knowledge; the task level represents task knowledge, it contains sets of tasks that are to be solved and methods solving them; the application level provides knowledge that is a combination of knowledge of the previous two levels depending on the task under consideration. Knowledge of all the levels is supposed to be described by means of common knowledge representation formalism.

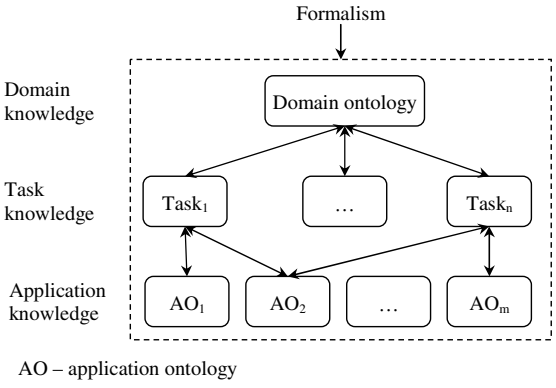


Fig. 1. Knowledge types

Within the framework domain knowledge is described by a domain ontology. Task knowledge represents sets of tasks and methods formalized by means of the common formalism. Domain knowledge that is used in the task solving in conjunction with the task knowledge make up an application ontology (AO). Every AO can represent knowledge involved in solving one or more tasks. The methodology includes the stages of task analysis, domain ontology building, task formalization, mapping specification, checking sufficiency of the domain ontology, AO composition, and AO consistency checking. For the proposed methodology the formalism of object-oriented constraint networks is used as the common formalism for ontology representation [10].

The mapping between domain and task knowledge is specified by mapping the methods' input and output parameters onto domain knowledge elements. By means of the accepted formalism the attributes of task knowledge represent methods' parameters. Thereby, the mapping is indicated by associative relations between attributes of task knowledge and attributes (in some cases, classes) of the domain ontology.

The mapping between domain and task knowledge is used to compose the AO by forming slices of the domain ontology. Domain knowledge that can serve as the method parameters is considered as relevant to the task. This relevant knowledge serves as basic knowledge or "seeds" for the slicing operation, which assembles knowledge related to the basic knowledge using an algorithm [e.g., 11; 12]. The resulting slice and the hierarchy of methods used in the task solving are integrated into AO that is validated and checked for the consistency.

3 Development of a Supply Chain Management Ontology

This section describes the application of the methodology to the supply chain management (SCM) area. The purpose of the methodology in this case consists in the building of a domain ontology for SCM and an AO representing a task knowledge.

The domain ontology was built by an expert. The main principle for the knowledge sources identification was the search for knowledge sources containing already developed and potentially reusable ontologies. An examination of many sources dealing

with the SCM domain did not reveal any ontologies of the domain in question. The ontology building was based on a definition of SCM, the postulate that SCM consists of supply chain and its components and management processes, the structures above, and an analysis of concepts used in publications on SCM problem.

Summing up definitions of SCM [e.g.,], it can be defined as management of flows of products and services, finances and information between different stages of supply chain from a supplier to a consumer / customer and managing operational activities of procurement and material releasing, transportation, manufacturing, warehousing and distribution, inventory control and management, demand and supply planning, order processing, production planning and scheduling, and customer service across a supply chain. The resulting SCM domain ontology is given in Fig. 2. The figure presents the class hierarchy for the classes of the taxonomy level following the root. SCM concepts are constructed to cover various supply chain stages, functions, decisions, and flows.

Taking into account the current tendency of the creation of an integrated supply chain that is the integration of separate supply chain stages based on the communication flow between the intermediate stages as well as the alignment of the goals of the intermediate stages with overall supply chain objectives [15] supply chain is considered running through different stages linked by material and information flows. The supply chain *stages* are [13;15]: supplier, manufacturer, transporters, distributors, and customer. Supply chain activities include *flow* of information, materials, and finances between different stages of a supply chain from suppliers to customer. *Information flow* includes capacity, promotion plans, delivery schedules, sales, orders, inventory, quality; *material flow* contains row materials, intermediate products, finished goods, material returns, repairs, servicing, recycling, disposal; *finances flow* is made up of credits, consignment, payments []. SCM is a mechanism to integrate supply chain functions taking place at the separate stages. Most of the *functions* happen within various stages, some of them cross the boundaries among several stages []. The functions operate on the supply chain flows.

Stages of the supply chain exchange the flows above through *orders*. *Orders* on *sale*, *replenishment*, *procurement*, *manufacturing*, *transportation*, or *customer order* can take place within every stage. An order within a certain stage is characterized by a

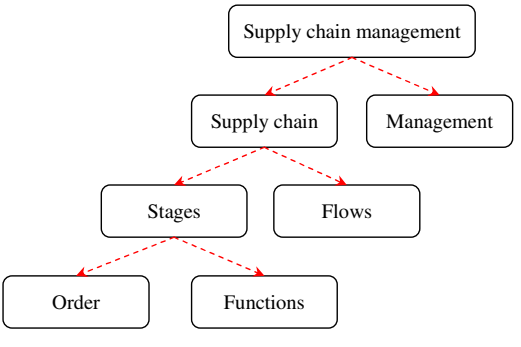


Fig. 2. Supply chain management domain ontology: top-level classes view

set of properties pertinent to orders of this stage and the order cycle representing different states of the order. *Management* concept refers to the coordination activities between supply chain components focusing on the planning and execution issues involved in managing the supply chain. These activities are represented as SCM tasks.

For the purpose of the paper the task of supply chain configuration has been chosen. Within the paper the configuration task is assumed to be parameterised by sector and specialisation as the input parameters, and time and cost as the output parameters. Sector describes an industry sector the supply chain belongs to; specialisation indicates what kind of production the supply chain deals with, e.g. mass production, mass customization, small branch production, piece production, etc. The parameters of time and cost are to return lead time of supply chain configuring and cost of the configuration respectively.

The aim of supply chain configuration problem is to find a feasible configuration with which the supply chain can achieve a high level of performance. Usually, there are two categories of configuration decisions: (1) structural decisions dealing with location, capacity, and distribution channel and (2) coordination decisions focusing on supplier selection, partnership, inventory ownership, sharing information about sales, demand forecast, production plan, and inventory [17]. Since the supply chain configuration problem is a complex task, only a part of an AO describing the task is given within the paper. The phase of bridging the whole domain ontology and supply chain configuration task is left out by the reason of a great number of concepts, attributes, and constraints of the domain ontology involved in the task. Partly the result of the bridging is illustrated within the AO in Fig. 3.

Part of this AO focuses on the supplier selection task as a subtask of logistics problem which is a part of the supply chain configuration problem. As a characteristic influencing supply chain performance supply chain cost is considered. In fact many cost items make up the total costs of the product required by the customer and the supply

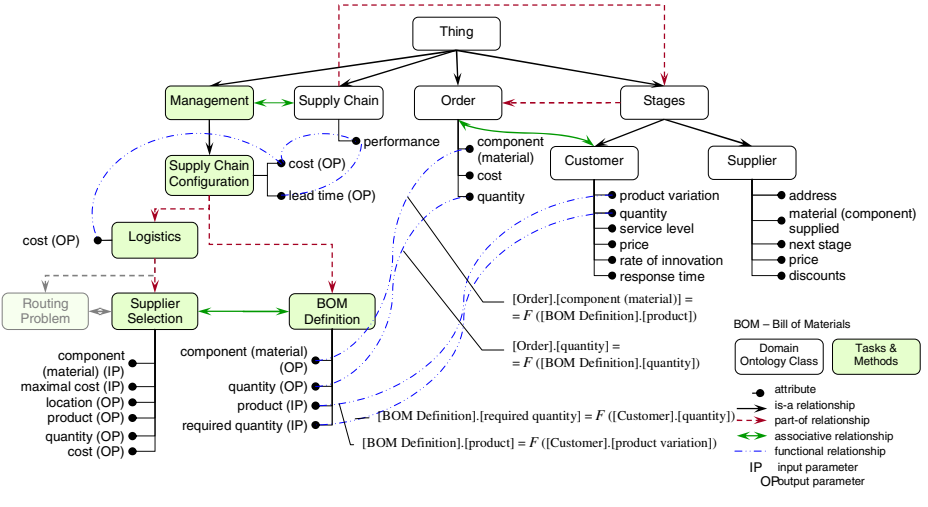


Fig. 3. AO (slice) for supply chain configuration problem

chain cost, among them manufacturing costs, shipping costs, and other are. This means that the complete AO includes all domain ontology classes that have an influence on supply chain cost. To simplify illustration interrelations between the domain ontology and the set of supply chain configuration tasks are given by the example of the task of forming order for bill of materials (BOM). This task defines a set of materials and components that compose the product ordered by the customer.

4 Matching SME Enterprise Ontology and SCM Domain Ontology

Matching companies to possible cooperation partners can use domain and individual enterprise ontologies. This requires a way to match each SME-ontology to the domain ontology to see what parts of the overall product or process a specific SME can provide, according to its ontology. The matching process should be performed using an appropriate method for combining ontologies.

According to some recent studies ([18] for example) there are two different approaches in the area of combining ontologies. Either the aim is to combine two ontologies of the same subject area, in which case it is called merging, or the aim is to combine two ontologies from different subject areas, in which case it is called integration. This is not a definition adopted by all researchers in the field but it is quite common and will be adopted in this paper. In a merging process the source ontologies are unified into a new ontology where it is difficult to determine which parts have been taken from which source ontology. Also in an integration process a new ontology is created, but here it is easy to see what parts come from the different source ontologies since they handle different subject areas. Fusion can be seen as a specific way to merge ontologies where the individual concepts loose or change their previous identity. In merging the concepts from the source ontologies are kept intact, but are combined to a new ontology. Finally there is the case when the result is not a new ontology but merely some kind of correspondences between the source ontologies. This process is here named alignment. It could be a first step in one of the above processes or it could be a sufficient result in itself. This process can be further divided into different categories such as mapping or finding translation rules between ontologies.

Among the different kinds of combination possibilities discussed, not primarily merging but integration is the most interesting concept, since the intention is to combine an enterprise ontology and a domain ontology. The whole process needs not necessarily to be performed either when forming a new ontology. It may be enough to just align the ontologies, for example by some kind of mapping.

With respect to manual approaches to combining ontologies, there is actually only one that stands out with its relative level of detail and well specified steps. This is the methodology from Pinto et al [19]. It is relatively detailed, as mentioned, but still not detailed enough to let it be performed without some expertise in ontology construction and management. In the field of semi-automatic approaches, the two probably most famous tools are Chimaera (see [20]) and the PROMPT suite (see [21]) for Protégé. These two tools both aim at merging ontologies by combining the concepts of the source ontologies, but there is nothing saying they could not also be used for integration if there are at least some small overlap between the ontologies.

Furthermore, there are some automatic approaches like the FCA-merge algorithm [22] or the IF-Map [23], which uses instances present in both ontologies as basis of the merging or mapping process. Similar algorithm can also be found in, e.g. [24], but they generally work without user intervention. Some approaches do not focus on the building of a new ontology but only of finding the right way to translate between ontologies, for example by articulation rules as in [25].

The analysis of methods for the ontology integration shows that these methods provide for different kind of resulting ontology generalization: from the alignment techniques where every ontology remains as it is to ontology fusion methods resulting in a unified ontology with concepts loosing or changing their identity.

It can be assumed that as a rule all the methods on ontology integration involve alignment as a first step [cf. 26]. Additionally, referring to the area of enterprise knowledge management, there is an opinion that imposing a single ontology on the enterprise is difficult if not impossible [27]. As a result, the authors consider combining distributed and heterogeneous ontologies by defining mappings between autonomous ontologies. In the case when a network of enterprises of different activities is under consideration obtaining a generalised ontology is even more difficult. According to the above the alignment techniques could be chosen as the first step. As ontologies evolve the correspondences between them could be analysed, possibly involving ontology learning practices. Based on obtained results other methods of the integration can be taken into consideration.

4.1 Domain Ontology Refinement

SME enterprise ontologies are highly minute. It provides full-dressed description of an enterprise, its business rules and solutions. This means that firstly, some knowledge contained in the SME ontology can represent the same concept more elaborately than the SCM ontology; and secondly, the SME ontology can provide the SCM ontology with new domain and task knowledge. These possibilities could be put into practice at the stage of integration of SCM and SME ontologies. The focus is on three situations when the domain ontology, or more specific the AO, has to be refined or enriched with new knowledge. The situations are selected to give an overview of possible variants that can arise at the integration phase. In fact every situation includes more variants resulting in AO refinement.

Situation a. Domain component refinement. The situation of the domain knowledge refinement arises when an SCM ontology and SME ontology represent the same concept by a different level of detail (Fig. 4). Taking into account the specific of the SCM ontology formalism: (i) the lack concepts can be directly included in an AO with appropriate modifications of the domains and constraints or (ii) a set of functional constraints describing rules how to handle inconsistencies between two representations can be added.

Decision on what kind of changes the SCM ontology needs is made relaying on an analysis of the requirements to the ontology and a study of the set of associative relations between domain and task knowledge incorporated into the AO. If the ontology is too general and knowledge provided by the SME ontology has a dramatic effect on the tasks related to this knowledge (e.g., the problem represented by AO as it is could

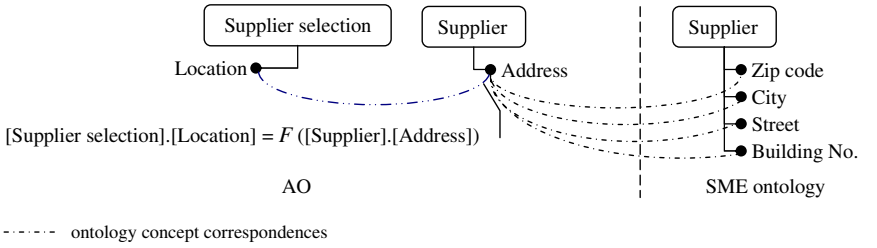


Fig. 4. Different level of detail

result in a wrong solution) then the SCM ontology has to be augmented with SME knowledge. This augmentation involves the phases peculiar to the middle-out approach. If SME knowledge does not produce any explicit contradictions and inconsistencies within AO then formulation and addition of functional constraints can be sufficient for AO refinement. So, the case illustrated in Fig. 4 does not require augmentation of the SCM ontology with all the address items since there is one-to-one mapping between the task parameter "location" and the attribute "address". Differences between the representations of the concept describing the supplier geographical position in the SCM and SME ontologies can be adjust by a set of rules.

Situation b. Task knowledge refinement. The situation when a revision of task knowledge is needed takes place when an SME ontology can provide an SCM ontology with new task knowledge. If the SME ontology provides a task method that an AO does not contain this method should be included into the AO with appropriate modifications of the task knowledge description, the associative relationships between domain and task knowledge, and the set of functional constraints. As in the previous situation, the inclusion of task knowledge into the AO produces a necessity of AO revision in accordance with the middle-out approach.

Situation c. Domain and task knowledge refinement. The situation when both domain and task components need a refinement occurs if an SME ontology provides new task knowledge but the domain component of an AO does not contain a complete set of the attributes serving as input parameters for task methods. In this case the SME ontology elements representing the new knowledge should be included into the AO. It corresponds to a revision and modification of the AO as proposed in the middle-out approach.

5 Conclusion

Agile and networked manufacturing require highly dynamic approaches of supply chain configuration. In particular for SME networks with temporary and geographically distributed members, both the capabilities of the member enterprises and the constraints and parameters of the relevant supply chain have to be modelled in a sufficiently precise way in order to support supply chain configuration adequately. SME enterprise ontologies in combination with a supply chain domain ontology are considered as an appropriate formalization and support technique for this task.

From a methodological viewpoint, ontology development has been investigated in numerous research activities, but is still a major concern in real-world projects. Especially in small-scale application contexts, development of an ontology still is not very common, as many companies hesitate to start this resource-intensive process. As a contribution to effort reduction, this paper proposes use and development of an SCM domain ontology that should be suitable for reuse and refinement in SME ontology projects for manufacturing industries.

Matching of ontologies in our application scenario is supposed to speed-up the process of finding the right enterprise partners for a given request in supply chain configuration. A suitable matching approach should ideally be either automatic or at least semi-automatic, in order to create significant advantages as compared to manually identifying the right enterprises. Thus, alignment of ontologies or finding correspondences only can be considered as the first step to more automated solutions.

An aspect requiring additional future investigation is the integration of enterprise and domain ontology not on a technical level of finding correspondences but on a knowledge refinement level. The paper introduced three situations, where an SME enterprise ontology could contribute knowledge to the SCM domain ontology: domain component refinement, task knowledge refinement, and domain and task knowledge refinement. Elaboration of these situations and analysis of their impact or contribution to further development of alignment approaches will be subject of future work.

References

1. Hieber R., Alard R. Supply Chain Management - A Survey of Practices and Trends in Swiss Industry. *Proc. IFIP WG 5.7 Working Conference*. Tromsö, Norway, 2000.
2. Sandkuhl K., Smirnov A., Henoch B. Towards Knowledge Logistics in Agile SME Networks - Technological and Organisational Concepts. In: Dolgui A., Soldek J., Zaikin O. (Eds.) *Supply chain optimisation: product/process design, facility location and flow control.* Kluwer Academic Publishers, ISBN 1-4020-8081-6.
3. Öhgren A., Sandkuhl K. Towards a Methodology for Ontology Development in Small and Medium-Sized Enterprises. In Guimarães N., Isaías P. (eds.) Proc. of the IADIS International Conference on Applied Computing, Algarve, Portugal, 2005, Vol. 1, 369-376.
4. Blomqvist E., Öhgren A., Sandkuhl K., Levashova T., Smirnov A. Formation of Enterprise Networks for Collaborative Engineering. Submitted to the CCE05 conference.
5. Staab S., Schnurr H.-P., Studer R., Sure Y. Knowledge Processes and Ontologies. *IEEE Intelligent Systems*, 2001, 16 (1). Special Issue on Knowledge Management, 26—34. URL:http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/isystems-knowledgeprocess.pdf.
6. Semantic Community Portals. *SWAD-E European Project*, HP Semantic Web Group, 2004, URL: http://www.hpl.hp.com/semweb/portal.htm.
7. Guarino N. Formal Ontology and Information Systems. *Proceedings of FOIS'98*. Trento, Italy. Amsterdam: IOS Press, 1998. 3—15.

8. P´erez A.G., Benjamins V.R. Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5*). Eds. by V.R.Benjamins, B.Chandrasekaran, A.Gomez-Perez, N.Guarino, M.Uschold. Stockholm, Sweden, 1999. URL: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/ Vol-18/.
9. On-To-Knowledge Methodology. *On-To-Knowledge: Content-driven Knowledge Management Tools through Evolving Ontologies*. Final Version, Deliverable 18, 2002. URL: http://www.ontoknowledge.org/downl/del18.pdf.
10. Smirnov A., Pashkin M., Chilov N., Levashova T. Haritatos F. Knowledge Source Network Configuration Approach to Knowledge Logistics. *International Journal of General Systems*, Taylor & Francis Group, 2003, 32 (3), 251—269.
11. Swartout B., Patil R., Knight K., Russ T. Toward Distributed Use of Large-Scale Ontologies. *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96).* Banff, Canada, 1996. http://www.isi.edu/isd/banff_paper/ Banff_ final_web/Banff_ 96_final_2.html.
12. Levashova T.V., Pashkin M.P., Shilov N.G., Smirnov A.V. Ontology Management. *Journal of Computer and System Sciences International,* Part II, 42 (5), 2003. 744--756.
13. Chopra S., Meindl P. Supply Chain Management. Strategy, Planning, and Operation, Prentice Hall, 2001.
14. Trent R.J. What Everyone Needs to Know about SCM. Supply Chain Management Review, March, 2004. URL: http://www.manufacturing.net/scm/index.asp? layout=rticle &articleID=CA409514.
15. Fully Integrated, Scalable Supply Chain Management Solution. *Solution Blueprints*, IBM Business Consulting Services, 2002. URL: http:// www. intel. com/business/bss/-solutions/blueprints/pdf/sb_ibmbcs0252.pdf.
16. Chopra S., Dougan D., Taylor G. B2B e-Commerce Opportunities. *Supply Chain Management Review*, May / June, 2001. 50—58.
17. Truong T.H., Azadivar F. Simulation Based Optimization for Supply Chain Configuration Design. *Proceedings of the 2003 Winter Simulation Conference* (S. Chick, P.J. Sánchez, D. Ferrin, and D.J. Morrice, eds.), 7—10 December, 2003, Vol. 2, 1268—1275. URL: http://ieeexplore.ieee.org/iel5/8912/28189/01261560. pdf.
18. Noy N.F., Musen M.A. Evaluating Ontology-Mapping Tools: Requirements and Experience. In: *Workshop on Evaluation of Ontology Tools at EKAW'02 (EON2002).* 2002.
19. Pinto H.S., Martins J. P. Ontology integration: How to perform the process. In: *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing*, 2001.
20. McGuinnes D.L., Fikes R., Rice J., Wilder S. An Environment for Merging and Testing Large Ontologies. In: *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, Breckenridge, 2000, 483—493.
21. Noy N.F., Musen M.A. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: *Proceedings of Seventeenth National Conference on Artificial Intelligence*, Austin, 2000, 450—455.
22. Stumme G., Maedche A. FCA-Merge: Bottom-Up Merging of Ontologies. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence, Seattle, USA, August 1-6 2001, San Francisco/CA*, Morgan Kauffmann, 2001.
23. Kalfoglou Y., Schorlemmer M.. IF-Map: An Ontology-Mapping Method based on Information-Flow Theory. *Journal of Data Semantics I*. Lecture Notes in Computer Science 2800, p 98-127, Springer, 2003.

24. Doan A., Madhavan J., Domingos P., Haleny A. Learning to Map between Ontologies on the Semantic Web. In: *11th International World Wide Web conference*, Honolulu, 2002.
25. Mitra P., Wiederhold G., Kersten M. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In: *Proceedings Conference on Extending Database Technology 2000 (EDBT'2000),* Konstanz, Germany, 2000.
26. Ding Y., Fensel D., Klein M., Omelayenko B. The Semantic Web: Yet Another Hip? *Data and Knowledge Engineering*, 41 (3), 2002, 205—227.
27. Maedche A., Motik B., Stojanovic L., Studer R., Volz R. Ontologies for Enterprise Knowledge Management. *IEEE Intelligent Systems*, March/April, 2003. 26—33.

Experiments Toward a Practical Implementation of an Intelligent Kanban System

James Z.M. Zhang[1], James Brusey[2,⋆], and Robert B. Johnston[1]

[1] Department of Information System, University of Melbourne,VIC 3000, Australia
z.zhang1@pgrad.unimelb.edu.au
robertj@unimelb.edu.au
[2] Institute for Manufacturing, University of Cambridge, CB2 1RX, UK
jpb54@cam.ac.uk

Abstract. This paper presents laboratory experiments to test a bottom up approach to production control and supply chain management. Built upon the successful traditional kanban (Card) system, the new intelligent system associates a kanban agent to each physical kanban. Instead of relying on demand forecast and planning, kanban agents reason about their own movements to adapt to changing demands. After previous simulations results of the intelligent system showed significant performance improvements over the traditional system, we further use the Auto-ID Laboratory at Cambridge University to test the feasibility of the idea in a realistic manufacturing environment. The results from the experiments demonstrated the superiority on several performance measures of the intelligent system compared to the traditional system used as a benchmark. Moreover, the implementation of the experiments exposed several real world constraints not shown in the simulation study and practical solutions were adopted to address these.

1 Introduction

Fast changing consumer demands require a manufacturer to respond in a prompt way to be able to remain competitive. A top down planning approach to supply chain management in this case is thus constantly faced with the need to reschedule. Moreover, with limited information access and bounded computational resources, centralised optimisation is either not feasible computationally or not economical to implement. These real world constraints in production and supply chain management motivated us to propose a bottom up approach to supply chain control, built over an industry strength system, the Just-In-Time kanban system[6]. The kanban system is a simple but effective card based reactive system, widely used in the highly competitive and globalised industries such as the automotive and the electronics sectors. The Traditional Kanban System (TKS) has proved to be very successful in production control and supply chain management. Its limitation lies in its inadequate adaptability to large demand fluctuations. In previous research using computer simulation, it was demonstrated that enhancing the dumb kanban system with basic reasoning ability results in a significant performance improvement over TKS [11]. However, as simulations inevitably reduce the possible state

⋆ J. Brusey is a member of the I*PROMS Network of Excellence.

V. Mařík, R.W. Brennan, M. Pěchouček (Eds.): HoloMAS 2005, LNAI 3593, pp. 257–268, 2005.

space it is not enough to guarantee the success of such an Intelligent Kanban System (IKS) in a real world application: a more realistic experiment might impose constraints not thought of in the simulation, thus exposing weaknesses in the system design. To overcome the inadequacy of simulation studies, the Auto-ID Laboratory at Cambridge University is employed as a test bed to test the idea of an intelligent kanban based system.

The Cambridge Auto-ID Laboratory was established to study the application of Auto-ID technology to flexible manufacturing system (FMS) and holonic manufacturing system (HMS). Previous works in the Auto-ID Laboratory (phase 1 and phase 2) have successfully demonstrated the feasibility and the advantage of applying the information provided by Auto-ID infrastructure to flexible manufacturing [2]. However, components replenishment processes were not considered in those works. It is understood that the components replenishment, and in general, other logistical processes, will benefit from the information provided by Auto-ID infrastructure. Nevertheless, in order to tap this potential there is a need to conduct quantified investigations into various ways of interpreting and utilising Auto-ID information to assist decision-making. Implementing a kanban based system in the Auto-ID Laboratory, therefore, could serve two purposes: explore an important application of Auto-ID technology and test the feasibility of IKS in an advanced realistic manufacturing environment.

Simulating kanban based systems in the Auto-ID Laboratory, the experiments described here aims to answer the following questions: a) Is it feasible to implement an IKS system in a realistic manufacturing environment such as the Auto-ID Laboratory? b) What is the effect of control strategy on the performance of the manufacturing system? c) Is the performance of IKS system significantly better than that of TKS system? d) What factors contribute to this performance difference, if any?

The rest of the paper proceeds as follows. Section 2 describes the processes of TKS and IKS in more detail. Section 3 discusses a proposal to realise the original kanban ideas in the context of the Laboratory and the software design implementing the proposal. Section 4 describes the experimental design and the test results from both TKS and IKS are analysed and compared. Section 5 discussed related works and finally conclusions are drawn in section 6.

2 The Kanban System

Physically, traditional kanbans are a set of cards circulating between neighbouring stages or echelons within a supply chain to authorise the release of orders [8]. In this system, the number of kanbans determines the total stock in an independent entity or echelon, and in normal situations, this number is fixed. Figure 1 shows in detail the working of a kanban system represented as a Petri net. The customer company (echelon) owns all the kanbans that circulate between the neighbouring customer and supplier echelons but not beyond. Taking a kanban (or token in terms of Petri net) from this group as an example, its life cycle is comprised of four parts: a) At stock point p_{2s}. The operation rule requires that any product that stays at the stock point must have a kanban attached to it. b) In transit upstream p_{2ut}. When the product associated with the kanban is pulled by another kanban from the customer's customer echelon (not shown), the detached kanban becomes free and travels upstream to the supplier company (typically

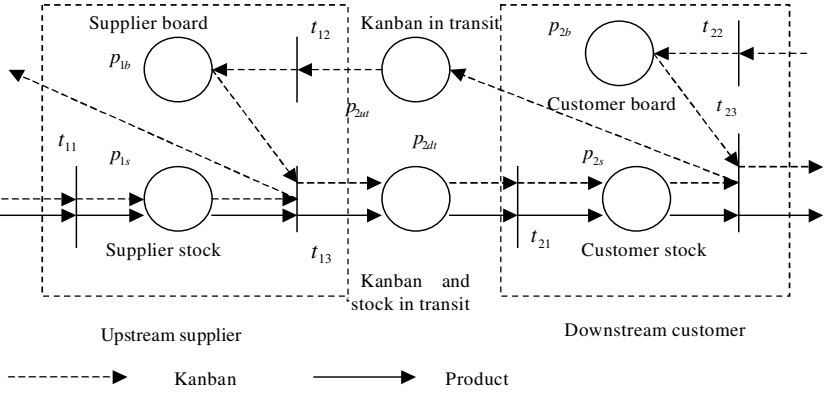


Fig. 1. A Petri net description of the operation of the traditional kanban system

with a returning delivery truck). This is a signal to the supplier to replenish stock. c) On kanban board p_{1b} . The free kanban will stay at the kanban board until transition t_{13} takes place, under the condition that a kanban with a corresponding product is available at stock point p_{1s}. d) In transit downstream p_{2dt}. When a product finally becomes available, it will be despatched downstream to the customer's site, with the kanban attached, while the kanban previously associated with that product will move back upstream to signal the need to replenish the used stock. Thus, the replenishment signals propagate upstream as products are shipped downstream.

In the kanban system, the release of orders is triggered by the actual consumption of goods instead of being determined by a planning process, so demand forecast and lead-time estimation are not essential. Simple but effective, the kanban system has proved its success in many competitive implementations around the world [9]. Its limitation is that it cannot cope with demand fluctuations larger than 10 percent [8]. One approach to large demand fluctuation is "Heijunka" or production smoothing, which aims to achieve a balanced workflow but still relies on demand forecast. Other proposals have been made to address the weakness by creating a system that adapts the number of kanbans to changing demand. The most widely used is optimisation search algorithms [7]. The problem is that they require a system go through a settling period before taking action again, thus timely response is not achieved.

To improve the tolerance to demand variations and enable a timely response to demand changes, we propose to adopt agent technology to endow reasoning capability to the originally dumb kanbans [6]. Situated right in operations environments, a physical kanban contains rich information about demand, supply, transport and even delivery failure. If a software agent (hereafter called a kanban agent) is assigned to each physical kanban, with the help of up-to-date Auto-ID information the kanban agent can have accurate near real time sensor data about the whereabouts of the physical kanban. If demand variation is reasonably low, the kanban will travel continuously without a hitch between neighbouring echelons to pull the needed product. Any significant disruption to this behaviour indicates insufficient capability of the bottom layer in regulating material flow, prompting intervention from the kanban agent with the goal of re-establishing

the free movement. The kanban agent, on its part, can then reason about the movements of its associated kanban (and possibly other kanbans if communication is included) and adjust the number of kanbans in circulation, the main decision variable in the system, in response to demand fluctuations, sensed as disruptions to the smooth movement of the physical kanban. The control decision is based on local information, without accessing global data about customer orders or availability of individual components. When demand consistently exceeds supply capacity such that more kanbans end up waiting in the kanban board of upstream supplier, an extra kanban could possibly be added to circulation to increase the supply capacity for that company. When demand is consistently below the supply level, stocks accumulate in stock point such that some kanbans stay at stock point for an extended time and it is possible to remove some kanbans out of circulation. In this way, the kanban number parameter at each echelon will be adjusted to adapt the overall system to long-term changes in demand and transport time.

It could be argued that the system is still reactive in nature. However, as an initial step toward a more "intelligent" system, the advantages of this agentised design are obvious. One advantage of this treatment is a greatly reduced reasoning burden for the upper agent layer by making full use of the capability of the reactive lower kanban layer. Secondly, because the proposed system is built upon a current TKS, such an implementation of IKS might allow agent-based control to be introduced over existing systems while minimising disruptions to the ongoing business operation. This characteristic is very important because it addresses some primary concerns industry people often have with the adoption of new agent based technology in manufacturing: ease of migration and compatibility with legacy systems. Thirdly, in the rare case of system failure of the agent-based intelligent system part, the supply chain can still rely on the underlying dumb kanban system to function instead of collapsing totally. The next section describes how these ideas were implemented in the Auto-ID Laboratory.

3 Design of the Experimental System

3.1 Experimental System Under the Semantics Context of Kanban

The Auto-ID Laboratory is comprised of a monorail transport sub-system, a set of self propelled shuttles travelling on the monorail, two material handling anthropomorphic robots, three docking stations, and two buffer stacks. The monorail transport system consists of a main loop running the length of the laboratory and two sub-loops connected to the main loop by control gates. At each strategic point, an Auto-ID reader picks up the Radio-Frequency (RF) signal and a program can identify and track the whereabouts of each moving part. Proposed by the Auto ID Centre, and later promoted by EPC Global, an Auto-ID system is comprised of three parts. Firstly, to uniquely identify a product instance, a 96-bit Electronic Product Code (EPC) is embedded in a RFID (Radio Frequency Identification) tag attached to the product instance. After the code information is captured by an RFID reader, an Object Name Service (ONS) system can translate this code to one or more Internet addresses from where detailed up-to-date information about the product instance can be retrieved, in the form of a XML vocabulary called Physical Markup Language (PML). As demonstrated in the Cambridge Auto-ID cell, a consumer can place an order for a customisable Gillette gift box and

Fig. 2. The empty box on the right hand side of this loop (sub-loop 1) corresponds to a kanban that is waiting to be fulfilled

the laboratory facility is capable of fulfilling that order without any human intervention with the help of Auto-ID information. The gift box may contain three out of a possible four Gillette products (namely razor, deodorant, gel, and foam, hereafter referred to as components), laid out in the box in a "T" shape configuration or in a row. For the purpose of our experiments, the existent laboratory facility is mapped conceptually to a two-echelon supply chain model comprising a manufacturer and its supplier. Upon receiving consumer orders for customised gift boxes, the manufacturer will try to find necessary resources. If not successful, it can send delivery shuttles to its supplier to pull the corresponding components.

In designing the experimental system, it was desirable to keep all the essential features of the original kanban system. However, due to the limitation of development resources, the correctness of the underlying algorithm has to take preferences over outward visibility. As a result, in the experiments the notion of virtual kanbans was used. Virtual kanbans corresponds to slots in a gift box or item positions in a stack. An empty slot in a gift box signals an unfulfilled demand and a slot with a component corresponds to a kanban with a product. Though this virtual kanban, in itself, cannot contain more information than the existence of demand, a local computer system can record the item detail of an order and the control algorithm can be built on this kind of information. Under these considerations, the experimental system works as follows. Sub-loop 1 serves the function of the kanban board for the invisible consumers. Representing consumer orders, a gift box with an empty slot could stay at sub-loop 1 indefinitely until a component of right type become available as shown in Fig. 2. When a component at stack 1 is used, the computer system will put a virtual kanban at stack 1, represented as a software object, which can be picked up by an incoming shuttle from the supplier.

When the system is running according to the principle of IKS, it works in a more flexible way than TKS in that a kanban agent associated with a virtual kanban can create a new virtual kanban for its supplier when it perceives the deficiency in supply capacity or destroy itself when it senses a redundancy of supply capacity. Notably, two parameters in the temporal domain $[t_k, t_s]$ are chosen to indicate the disruption of the desired behaviours. t_k is the waiting time of a kanban on a kanban board, to measure the unfulfilled demand, and t_s is the idling time of a kanban at a stock point, to measure the excess

supply. The two variables are monitored by a kanban agent and transgression over the preset limits $[T_k, T_s]$ will trigger the adjustment process. For example, if a kanban stays at a stock point for too long, this will indicate a need to remove the stock, together with the kanban. If demand is consistently larger than available stock, the waiting time of an incoming kanban in a kanban board will accumulate, to the extent of triggering control decision to increase the number of kanban in upstream company. The algorithm works as follows:

$$\begin{cases} t_k \geq T_k, \text{ duplicate} \\ t_s \geq T_s, \text{ demolish} \end{cases}$$

3.2 Design of the Software

For the software design, the challenge was how to better program reactivity to improve the system efficiency. The key is to avoid premature and over commitment to a plan, which usually assumes the normal state of affairs is for things to go according to a plan, and disruption is unlikely [1]. Viewed from the component level, the Laboratory environment, however, teems with unpredictability arising from complex interactions, which can easily render a plan irrelevant or sub-optimal when it is actually executed. In a previous design [5], specific components are logically assigned to an order the moment it is presented to the system. Though availability of the right components is ensured when a shuttle arrives at a dock, it is observed that this treatment is susceptible to disruption during the course of the plan, and more importantly, it may fail to take

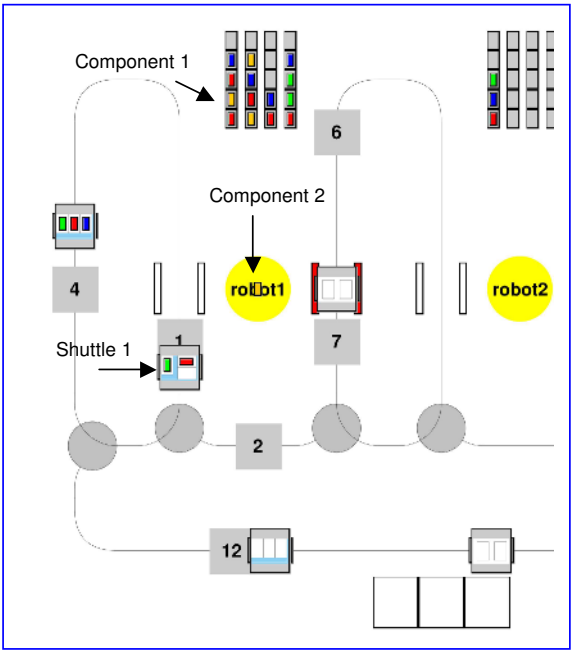


Fig. 3. A situation where an opportunistic decision seems better

some opportunities that could arise through the interaction of an agent and its surrounding environment.

For example, in Fig. 3, a shuttle (near label 1, referred to as shuttle 1) is about to enter the dock station near robot 1 in need of a yellow component. The yellow component (component 1) at stack 1 has been allocated for this purpose by the planning process. At this time robot 1 just grabs a yellow component (component 2) from the right dock station. It is obvious the system would perform better if robot 1 puts component 2 directly into the incoming shuttle instead of getting component 1 from the stack. The working mechanism of the kanban system can achieve this "Just-In-Time" association without planning beforehand. The time saving could be quite considerable because the stack operation works like a queue: a robot can only grab the bottom component. Therefore, if a robot wants to grab a component in the middle of a stack, it has to firstly pull out all the components below that component and replace them into the stack one by one. This example highlights the subtlety of reactive programming.

4 The Experiments

4.1 System Input and Output

The input of the system is consumer orders, generated automatically into a waiting queue. The content of an order includes a box type, and three component types out of a possible four. The whole test time is divided into three periods of equal length. In each period, the demand distribution remains the same. Starting from a consumer demand evenly distributed to the four components types, after each third of test time, the proportion among components types is manipulated as shown in Table 1. As can be seen from Table 1, in period 2 demands for deodorants and gels are twice those for razors and foams and when the experiments proceed to period 3, the demands for deodorants and gels are only one third of that for razor, and one half of that for foam. By imposing a simple demand distribution like this, we can observe the adaptabilities of both systems to large demand fluctuation. The order releasing sequences are designed to be the same for tests of both systems to ensure the integrity of comparisons.

The outputs of the system are selected performance indicators. We use order wait time to measure the quality of customer service, defined as the time from when an order is placed by customers to the time a product is delivered. The second performance indicator, total stock, in this context includes stocks at a stock point and products in transportation. The third performance indicator, the rate of back orders, is the ratio of the number of delayed orders to that of all arriving orders.

Table 1. The relative proportions among different components types during the experimental run

Component type	The first 1/3 time	The second 1/3 time	The Third 1/3 time
Deodorant	25%	33%	14.3%
Gel	25%	33%	14.3%
Razor	25%	16.7%	42.9%
Foam	25%	16.7%	28.6%

4.2 Running of the Experiments

Six shuttles are used in the experiments, three of which are allocated to carry gift boxes and another three are used as component carriers. Four components of each type are required. Before the start of the experiment, all components are put in the two stacks and all the six shuttles stay before the stop sign along the main loop. The experiment ran for about one hour. To run the experiment simulating IKS, all the materials, equipment and their placements are the same as those of TKS. The difference is that in the computer running the program, the program type is changed from TKS to IKS in a configuration file. The data from the initial 3 minutes (about 5% of total experiment time) are discarded to remove the transient effect. The 3-minute settlement period was determined by the analysis of data from several trial runs.

4.3 Results Discussion

Based on experimental results from the ten test runs, the whole experiment can be divided into three phases in terms of order wait time. Up to the first three orders, the order wait times for both systems are roughly the same because initial stocks can be readily used to fulfil the first batch of orders. After the initial stocks run out, the differences in replenishment strategy have a strong effect on order wait time. In this phase, it can be seen IKS performs much better than TKS in terms of both mean and variance of order wait time. In the last phase starting from the 11th completed order, only 1 TKS fulfilled 12 orders. In contrast, all 5 runs of IKS completed 12 orders and two of them completed 13 orders.

According to this analysis, it will be interesting to look into the difference between the two systems from order 4 to order 10 as shown in Fig. 4. After the initial stocks are used, the following boxes have to wait in sub-loop 1 for some time until they are fully packed. According to Table 1, in phase two, the demands for deodorant and gel are twice those of razor and foam. Therefore, it is more likely the slots for a deodorant or gel will remain empty at sub-loop 1 than those for a razor or foam. In accordance with the IKS algorithm, the system could create more virtual kanbans for deodorant and gel. On the other hand, the stocks for razor and foam, if any, are likely to stay at the stock point for an extended time. Once they get used, the virtual kanbans previously associated with them could be removed from circulation. Through this mechanism, IKS can adjust the number of kanbans for each component type to reflect the actual demand situations, realising an “optimal” use of system resources such as transportation shuttles, docks and packing robots, without making demand forecasts.

The second performance indicator is total inventory. Fig. 5 shows a comparison of total inventory through the test period. The horizontal axis is the opening time of the observation window, that is, the time to “sample” the system. The vertical value for each point is the mean of total inventories of all 5 test runs for either TKS or IKS. Because IKS completed more consumer orders in the same time interval, more components existed in the form of finished orders rather than inventories at stock points. For both systems, initially the total inventory is high, because all the components stay at the stock point.After components are packed into gift boxes, the total inventory decreased as a result. Afterwards there is a slight build-up in inventory in TKS, because

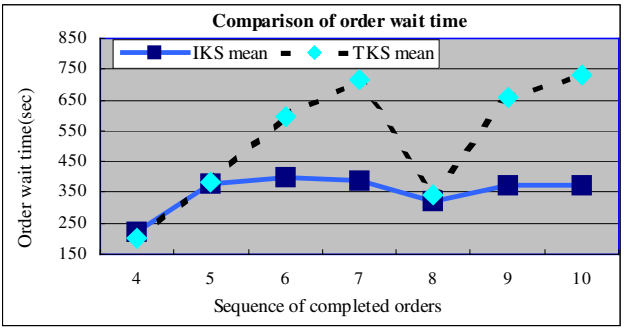


Fig. 4. Comparison of order wait time

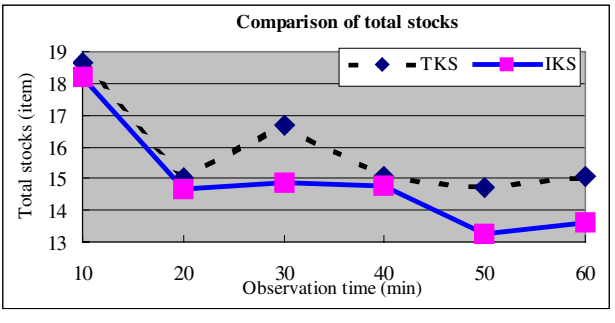


Fig. 5. Comparison of total stocks

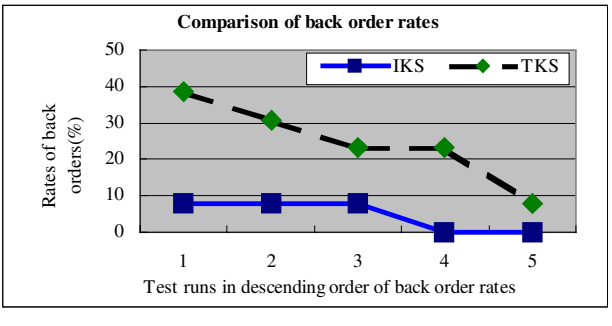


Fig. 6. Comparison of rates of back order

type mismatch between stocks and demand is inevitable when demand changes while the Kanban number for each type remains fixed.

The third performance indicator, the rate of back orders, can be easily obtained using the formula: rate of back order = (total orders - completed orders)/total orders. Because the number of all generated orders is very large, in the interest of clarity, the maximum number of all completed orders is used as a denominator. The results thus calculated are shown in Fig. 6 in descending order of the rates of back orders.

4.4 Validity of the Results

Statistical tests are used to verify whether there are significant differences in performances between TKS and IKS. Unpaired t-test for order wait time shows the p value to be 0.17 per cent, so it is concluded that IKS does use shorter time to complete an order than TKS in average. Unpaired t-test for back order rates shows that the p value is 1.4 per cent, and it is concluded that IKS has significantly lower back order rate. For the standard deviation of wait time, the unpaired t test shows the p value is only 0.027 per cent. The conclusion is that IKS fulfils orders in a more reliable way with less volatility. For total inventories, as observations are made at the same time points for both systems, the results are paired. Given p value of 1.6 per cent from paired t-test, it is concluded that IKS maintains lower total inventories than TKS.

5 Related Work

A kanban agent, representing an order, is roughly equivalent to an order holon in PROSA architecture [10] or a WIP agent in HCBA architecture [3]. However, they are different in two important aspects. First, the kanban process is repetitive in nature instead of one-off, giving rise to the emergence of patterns and a possibility of continuous improvement through learning. Situated in a local environment between two neighbouring stages, the kanban agent can monitor the effect of its action on the system performance and improve its decision making in the next round of interaction with the environment. Secondly, while both the order holon and the WIP agent will travel across stage boundaries until an order or a product is completed, the kanban agent relies only on information about itself, and consequently, the communication overhead is greatly reduced. Nevertheless, we acknowledge the myopic nature of the system might not always reach optimal global performance and in future, some form of inter-agent communication may be necessary.

From a pure algorithm point of view, our work is most related to biologically inspired systems like the routing wasp agents in a painting shop [4]. The response thresholds to stimulus for the wasp agents correspond to waiting time limits for the kanban agents. The difference is that while in both wasp systems and market based systems decisions have to be made by comparing the different bids or thresholds from a cluster of agents, in our system the decision to adjust kanban number is based solely on the discretion of an individual agent. This streamlined design is viable because the waiting time limits result from the interaction between all local kanban agents and the environments, reflecting the actual situations of demand and supply. Therefore, a kanban agent knows indirectly the effects of actions other kanban agents take, in the form of delays to the smooth movement of physical kanbans.

6 Conclusions

In this paper, we have presented laboratory experiments to test the idea of the Intelligent Kanban System. To summarise the findings: a) It is feasible and practicable to implement IKS in an Auto-ID enabled environment. b) When all performance indicators are

considered simultaneously, it is shown that the IKS control significantly improves system performance. c) IKS improves the customer service level by producing a significant reduction in order wait time. Moreover, it was found to reduce total inventories in the Laboratory environment. d) The reason for this significant performance improvement lies in the capability of IKS to "optimise" the allocation of system resources in response to demand change, through reasoning about its own behaviour, instead of doing demand forecast and planning.

The design of the system addresses two important concerns manufacturing industry often has with the agent based technology: ease of migration and compatibility with legacy systems. By building upon a proven industry control system, an implementation of IKS allows agent-based control concepts to be introduced over existing systems while minimising disruptions to the ongoing operation of business. It is also very robust to uncertainty . In the rare case of system failure of the agent-based intelligent system part, the supply chain can still rely on the underlying dumb kanban system to function instead of collapsing totally. Compared to top-down planning systems such as MRPII, the bottom up Intelligent Kanban System remains very promising: the benefits are evident and predictable and the risks are effectively constrained.

After initial simulation results suggested the feasibility of the idea of IKS, the Auto-ID Laboratory experiments further demonstrate the practicability of the idea in a realistic manufacturing environment. Testing in such a realistic environment has shown several issues not discovered in simulation and practical solutions have been adopted to address them accordingly. With evidence gained from the experiments, we have more confidence to believe the benefits obtained in the laboratory can be readily reproduced in real life scenarios.

Acknowledgments

This work is supported by Australian Research Council Linkage Grant LP0211. We acknowledge the contribution of funds and time by Agent Oriented Software Pty. Ltd. to this project. Also, we thank Cambridge Auto-ID Laboratory for providing the excellent facility to carry out our experiments. In particular, we are indebted to Allan Thorny, Dr. Mark Harrison, Andy Shaw, Dr. Duncan Macfarlane and PhD students Ajith Parlikad, Anand Kulkarni, C Y Wong and Amro Farid from IfM at Cambridge University for their continuous supports and many inspiring discussions. We are also grateful to anonymous reviewers for their valuable suggestions in improving the paper.

References

1. Agre, P. E. and Chapman, D.: Pengi: An implementation of a theory of activity, The Sixth National Conference on Artificial Intelligence (AAAI-87), Seattle, WA, USA, 1987
2. Brusey, J., Fletcher, M., Harrison, M., Thorne, A., Hodges, S. and McFarlane, D.: Auto-ID Based Control Demonstration - Phase 2: Pick and Place Packing with Holonic Control, Institute for Manufacturing, Cambridge University, 2002
3. Chirn, J. and McFarlane, D.:Application of the Holonic Component-Based Approach to the Control of a Robot Assembly Cell, IEEE Conference on Robotics and Automation, San Fransisco,2000

4. Cicirello, V. and Smith, S.: Wasp-like Agents for Distributed Factory Coordination, Robotics Institute, Carnegie Mellon University,2001
5. Fletcher, M. and Brusey, J.: The Story of the Holonic Packing Cell, 2nd Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia, 2003
6. Johnston, R. B., Wallis, P., Zhang, J. and Jones, R.: Reasoning about activity: robots, kanbans and intelligent infrastructure, AAMAS 03, the fifth international workshop on agent-oriented information systems, Melbourne, Australia, 2003
7. Martin, A. D., Chang, T. M.,Yih, Y. and Kincaid, R. K.: Using tabu search to determine the number of kanbans and lotsizes in a generic kanban system, Annals of Operations Research, vol. 78, pp. 201-217, 1998.
8. Monden, Y.: Toyota production system : an integrated approach to Just-In-Time, Norcross, Ga., Engineering & Management Press, 1997
9. Schonberger, R. J.: Japanese manufacturing techniques : nine hidden lessons in simplicity, New York, Free Press, 1982
10. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L. and Peeters, P.: Reference architecture for holonic manufacturing systems: PROSA, Computers in Industry, Vol 37, 3, 1998
11. Zhang, J., Wallis, P. and Johnston, R. B.: Intelligent Kanban: Evaluation of a Supply Chain MAS Application Using Benchmarking, 2005 IEEE International Conference on E-Technology, E-Commerce and E-Service, Hong Kong, IEEE Computer Society press, 2005

Author Index