Vladimír Mařík
Thomas Strasser
Alois Zoitl (Eds.)

LNAI 5696

Holonic and Multi-Agent Systems for Manufacturing

4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2009
Linz, Austria, August/September 2009, Proceedings

Springer

Vladimír Mařík
Thomas Strasser   Alois Zoitl (Eds.)

Holonic and Multi-Agent Systems for Manufacturing

4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2009
Linz, Austria, August 31 - September 2, 2009
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Vladimír Mařík
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
Technická 2, 16627 Praha 6, Czech Republic
and Rockwell Automation Research Center Prague
Pekařská 10a/695, 15500 Prague 5, Czech Republic
E-mail: marik@labe.felk.cvut.cz

Thomas Strasser
PROFACTOR GmbH
Im Stadtgut A2, 4407 Steyr-Gleink, Austria
E-mail: thomas.strasser@profactor.at

Alois Zoitl
Vienna University of Technology
Automation and Control Institute
Karlsplatz 13, 1040 Vienna, Austria
E-mail: zoitl@acin.tuwien.ac.at

Library of Congress Control Number: 2009932881

CR Subject Classification (1998): I.2.11, J.6, I.6, D.2, D.2.9

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-03666-X Springer Berlin Heidelberg New York
ISBN-13 978-3-642-03666-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12730490 06/3180 5 4 3 2 1 0

Preface

The research of holonic and agent-based systems is developing very rapidly. The community around this R&D topic is also growing fast - despite the fact that the real-life practical implementations of such systems are still surprisingly rare. However, the managers in different branches of industry feel that the holonic and agent-based systems represent the only way of managing and controlling very complex, highly distributed systems exploring vast volumes of accumulated knowledge. The relevant research and development activities gain more and more visible support from both industry as well as public sectors. Quite naturally, the number of scientific events aimed at the subject field is also growing rapidly. We see new lines of conferences like INDIN, we observe a strong focus of the already well-established conferences, e.g., INCOM or ETFA, being shifted toward holonic and agent-based manufacturing systems. We see an increased interest of the IEEE System, Man and Cybernetics Society, especially its Technical Committee on Distributed Intelligent Systems which leverages the experience gathered by the members of the former Holonic Manufacuting Systems (HMS) consortium. We see a clear orientation of the IEEE SMC Transactions, part C, toward applications of agent-oriented solutions. The same is true of the *International Journal on Autonomous Agents and Multi-Agent Systems* (JAAMAS). This is a really good sign of the increasing importance of the field.

We are convinced that it is important to continue the HoloMAS events that belonged to the pioneering melting pots for ideas connected with distributed decision making and control in industry and which have already gained an international reputation. The first six HoloMAS events held under the DEXA-event umbrella (three workshops, 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, the Second HoloMAS 2005 held in Copenhagen, and the Third HoloMAS 2007 held in Regensburg) helped to bring together the research communities focused at agent-based industrial solutions, to discuss 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. For HoloMAS 2009, we would like to document the trends and progress in the subject field. The focus of papers reaches from motivation papers on self-organization of holonic systems, description of agent activities as services exploring specific service-oriented architectures, and discussions on practical usability of agent-oriented solutions to such topics as knowledge-centerd approaches and holonic manufacturing systems up to practical applications of agent-oriented systems.

We have decided to organize the HoloMAS conference bi-yearly, on even years, under the DEXA umbrella and to focus the attention to specific IEEE SMC events on the odd years. An IEEE DIS workshop with a special track covering the "obvious" HoloMAS topics was organized in Prague in June 2006. Similarly, the IEEE SMC Conference on Distributed Human–Machine Systems (DHMS 2008), which has absorbed the HoloMAS field, was held in Athens, Greece in March 2008 (http://www.action-m.com/dhms2008/).

This approach allows the HoloMAS community to be better integrated with both the information society-oriented DEXA community as well as the IEEE Society aimed at human–machine systems and cybernetics.

We are happy to announce that the HoloMAS 2009 conference was held again under the technical co-sponsorship of the IEEE SMC Society in cooperation with the I*PROMS EU Network of Excellence.

The I*PROMS EU Network of Excellence http://www.iproms.org is aimed mainly at research and development of advanced systems for manufacturing and control. Holonic and agent-based solutions represent one of the latest trends in this area and are in the focus of the I*PROMS activities.

There were 47 papers submitted, prepared by members of academia as well as industry worldwide. The PC chose 31 papers to be presented and included in this volume. They contain representative results of the corresponding research and provide an outstanding overview of what is the current state of the art.

Moreover, there were three invited talks specially tailored for HoloMAS 2009, namely:

P. Leitão: Holonic Rationale and Self-Organization on Design of Complex Evolvable Systems (short version of the paper included in this volume)

E. Adam, P. Valckenaers, P. Leitão: Robustness and Cooperation in Holonic Multi-Agent Systems

M. Luck: Flexible Behavior Regulation in Agent-Based Systems

The invited speeches will be published later as full papers in the newly established *Journal on Transactions on Large-Scale Data and Knowledge-Centered Systems* by Springer.

The HoloMAS 2009 conference created an excellent, highly motivating environment, and helped to continue integration of the community. It contributed to a better 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 display window of the holonic and agent-based manufacturing research offering information on 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 and to Gabriela Wagner for all her organizational efforts which were of key importance for the success of this event.

We would like to thank the IEEE SMC Society for its technical co-sponsorship as well as the I*PROMS EU Network for their support and cooperation.

June 2009

Vladimír Mařík
Thomas Strasser
Alois Zoitl

Organization

Co-chairs

Vladimír Mařík	Czech Technical University in Prague and Rockwell Automation, Czech Republic
Thomas Strasser	Profactor, Austria
Monique Calisti	Whitestein Technologies, Switzerland

Program Committee

José Barata	Universidade Nova de Lisboa, Portugal
Vicente Botti	Universidad Politecnica de Valencia, Spain
Jeff Bradshaw	University of West Florida, USA
Robert W. Brennan	University of Calgary, Canada
Birgit Burmeister	Daimler AG, Germany
Luis M. Camarinha-Matos	Universidade Nova de Lisboa, Portugal
Marco Colla	SUPSI/ICIMSI, Switzerland
Armando W. Colombo	Schneider Electric, Germany
Luca Ferrarini	Politecnico di Milano, Italia
Motohisa Funabashi	Hitachi, Japan
Dominic Greenwood	Whitestein Technologies, Switzerland
William Gruver	Simon Fraser University, Canada
Kenneth Hall	Rockwell Automation, USA
Agirre Ibarbia	Fundacion Fatronik, Spain
Toshiya Kaihara	Kobe University, Japan
Kari Koskinnen	Helsinki University of Technology, Finland
Dilip Kotak	Simon Fraser University, Canada
Jose L.M. Lastra	Tampere University of Technology, Finland
Paulo Leitao	Polytechnic Institute of Braganca, Portugal
Francisco Maturana	Rockwell Automation, USA
Duncan McFarlane	Cambridge University, UK
Michal Pěchouček	Czech Technical University in Prague, Czech Republic
Duc Pham	Cardiff University, UK
Milan Rollo	Czech Technical University in Prague, Czech Republic
Martijn Rooker	Profactor, Austria
Stuart Rubin	SPAWAR Systems Center, San Diego, USA
Ilkka Seilonen	Helsinki University of Technology, Finland
Leonid Sheremetov	Mexican Oil Institute, Mexico
Alexander Smirnov	SPIIRAS, Russia
Paul Valckenaers	Katholieke Universiteit, Leuven, Belgium
Antonio Valentini	OOONEIDA, Canada

Tomáš Vlček	Czech Technical University in Prague, Czech Republic
Pavel Vrba	Rockwell Automation, Czech Republic
Valeriy Vyatkin	University of Auckland, New Zealand
Alois Zoitl	Vienna University of Technology, Austria

External Reviewers

Jan Bezdíček	Rockwell Automation, Czech Republic
Michal Jakob	Czech Technical University in Prague, Czech Republic
Petr Štěpán	Czech Technical University in Prague, Czech Republic
Pavel Tichý	Rockwell Automation, Czech Republic
Jiří Vokřínek	Czech Technical University in Prague, Czech Republic

Organizing Committee

Petr Benda	Czech Technical University in Prague, Czech Republic
Barbora Jeníková	Czech Technical University in Prague, Czech Republic
Gabriela Wagner	FAW, University of Linz, Austria

Table of Contents

Holonic Rationale and Self-organization on Design of Complex Evolvable Systems

Paulo Leitão

Polytechnic Institute of Bragança, Campus Sta Apolonia, Apartado 1134,
5301-857 Bragança, Portugal
pleitao@ipb.pt

Abstract. Old-fashioned centralized and rigid control structures are becoming inflexible to address the current requirements of reconfigurability, responsiveness and robustness. The Holonic Manufacturing Systems (HMS) paradigm, which was pointed out to face these requirements, translates the concepts inherited from social organizations and biology to the manufacturing world. It offers an alternative way of designing adaptive systems where the traditional centralized control is replaced by decentralization over distributed entities. In spite of its potential, methods regarding the self-adaptation and self-organization of complex systems are still missing. This paper discusses how the insights from biology in connection with new fields of computer science can be useful to enhance the holonic design aiming to achieve more self-adaptive and evolvable systems.

Keywords: Holonic Manufacturing Systems, Biological-inspired Theories, Emergent behavior, Self-organization.

1 Introduction

Global markets and customer demands are imposing strong requirements to manufacturing companies, namely in terms of flexibility, re-configurability, responsiveness and robustness. Re-configurability, that is the ability of the system to dynamically change its configuration, usually to respond to dynamic changes in its environment, assumes a particular role in the new generation of production control systems, providing the way to achieve a rapid and adaptive response to change, which is a key enabler of competitiveness.

Traditional manufacturing control approaches typically fall into large monolithic and centralized systems, exhibiting low capacity of adaptation to the dynamic changes of their environment and thus not supporting efficiently the demanded requirements. The quest for re-configurability requires a new class of intelligent and distributed manufacturing control systems that operates in a totally differentiated way when compared with the traditional ones, being the centralized and rigid control structures replaced by the decentralization of control functions over distributed entities. Multi-agent systems [1], derived from distributed artificial intelligence and based on the ideas proposed by M. Minsky [2], suggest the definition of distributed control based on autonomous agents that account for the realization of efficient, flexible, reconfigurable and robust overall plant control, without any need for centralized control.

Other emergent manufacturing paradigms have been proposed during the last years, namely Bionic Manufacturing Systems (BMS) [3] and Holonic Manufacturing Systems (HMS) [4], which uses biological and social organization theories as sources of inspiration. These paradigms are unified in proposing distributed, autonomous and adaptive manufacturing systems. Among others, the Product-Resource-Order-Staff Architecture (PROSA) [4], the ADAptive holonic COntrol aRchitecture for distributed manufacturing systems (ADACOR) [5], the Holonic Component-Based Architecture (HCBA) [6] and the P+2000 [7]), are successful examples of the application of the HMS principles.

The application of multi-agent systems and holonic paradigms, by themselves, does not solve completely the current manufacturing problems, being necessary to combine them with mechanisms to support the dynamic structure re-configuration, thus dealing more effectively with unpredicted disturbances and minimizing their effects. In other words, questions like how the global production optimization is achieved in decentralized systems, how temporary hierarchies are dynamically formed, evolved and removed, how individual smart components self-organize and evolve to support evolution and emergence, and how to adapt their emergent behavior using learning algorithms, are yet far from being answered. In fact, in spite of the interesting potential introduced by these paradigms, methods and tools to facilitate their design and maintenance, in particular regarding to their self-adaptation and self-organization properties, are required.

Biology and nature seem suitable sources of inspiration to answer the above questions, achieving better solutions for self-adaptive and evolvable complex systems. A recent article published in the National Geographic magazine reinforces this idea, stating that "*the study of swarm intelligence is providing insights that can help humans to manage complex systems*" based on the idea that "*a single ant or bee isn´t smart but their colonies are*" [8]. In this context, this paper discusses the benefits that the bio-inspired theories can bring to the manufacturing world, analyzing how the insights from biology, such as emergence and self-organization, in connection with new fields of computer science, such as artificial life and evolutionary computing, can be applied to power the design of holonic systems aiming to achieve more adaptive and re-configurable systems. The ADACOR holonic approach is used to illustrate the application of some biological inspired theories to design an adaptive production control system that evolves dynamically between a more hierarchical and a more heterarchical control architectures, based in the self-organization concept. Other examples that can be named are the introduction of an ant technique as coordination mechanism in the PROSA architecture [9] and Air Liquide that uses an ant-based strategy to manage the truck routes for delivering industrial and medical gases [8].

The rest of the paper is organized as follows: Section 2 overviews the holonic rationale and Section 3 illustrates the use of biological theories to build complex systems, namely the emergence and the swarm intelligence principles. Section 4 discusses how self-organization principles can be applied to achieve self-adaptive, re-configurable and evolvable complex systems, and Section 5 presents the ADACOR example of applying biological inspired theories to achieve an adaptive production control system. Finally, Section 6 rounds up the paper with the conclusions.

2 Backing to the Holonic Rationale

Manufacturing environment is usually characterized by being:

- *Non-linear*, since it is based on processes that are regulated by non-linear equations where effects are not proportional to causes.
- *Complex*, since the parameters that regulate those processes are interdependent: when changing one, the others are changed, resulting in instability and unpredictable systems.
- *Chaotic*, since some processes work as amplifiers, i.e. small causes may provoke large effects. For example, the occurrence of a small disturbance in a machine may affect the system's productivity.

As result, manufacturing systems are usually unpredictable and very difficult to control. Additionally, manufacturing is constantly subject to pressures from market that demands for customized products at shorter delivery time, which requires the ability to respond and adapt promptly and efficiently to changes.

Having in mind the particularities of the manufacturing domain, suitable paradigms are required to address this challenge. HMS is a paradigm that translates the concepts developed by A. Koestler into a set of appropriate concepts for manufacturing domain. Koestler introduced the word holon to describe a basic unit of organization in living organisms and social organizations [10], based on H. Simon theories and on his observations. Simon observed that complex systems are hierarchical systems formed by intermediate stable forms, which do not exist as auto-sufficient and non-interactive elements but, on the contrary, they are simultaneously a part and a whole. The word holon is the representation of this hybrid nature, being a combination of the Greek word holos, which means whole, and the suffix on, which means particle.

The HMS is a holarchy that integrates the entire range of manufacturing activities, i.e. a society of holons that can co-operate to achieve a goal. A holon is an autonomous and co-operative entity that can represent a physical or logical activity, such as a robot, a machine or an order. In HMS, the holons behaviors and activities are determined through the cooperation with other holons, in opposition of being determined by a centralized mechanism. Considering the Janus effect, being simultaneously self-contained wholes to their subordinated parts and dependent parts when seen from the higher levels, it is possible to step-wise decompose a holon into several others holons, allowing the reduction of the problem complexity. An example, a holon representing a manufacturing cell is simultaneously the whole, encapsulating holons representing the cell resources, and the part, when considering the shop floor system.

At the moment, the industrial adoption of these approaches has fallen short of expectations, and the implemented functionalities are normally restricting [11]. Additionally, a missing issue in the design of holonic systems is the application of self-adaptation and self-organization properties that causes systems to become increasingly more reconfigurable, organized and efficient. The challenge is to go back to the foundations of these theories and combine them with biological inspired theories to develop more adaptive and evolvable systems, which can be easily deployed into real environments. For this purpose, biology and nature provide a plenty of mechanisms that can be applied, as illustrated in the following sections.

3 Swarm to Achieve Emergent Complex System Behavior

Biology and nature offer powerful mechanisms, refined by millions of years of evolution, to handle emergent and evolvable environments. In nature, complex systems are built upon entities that exhibit simple behaviors and have reduced cognitive abilities, where a small number of rules can generate systems of surprising complexity [12]. As example, an ant or a bee present simple behavior but their colonies exhibit a smart and complex behavior.

The emergence concept reflects this phenomenon and defines the way complex systems arise out from a multiplicity of interactions among entities exhibiting simple behavior. The emergent behavior occurs without the guidance of a central entity and only when the resulted behavior of the whole is greater and much more complex than the sum of the behaviors of its parts [12].

Swarm intelligence, also inherited from biology, extends this behavior, being defined as the emergent collective intelligence of groups of simple and single entities [13]. In fact, swarm intelligence is typically made up of a community of simple entities, following very simple rules, interacting locally with each another and with their environment. It offers an alternative way of designing intelligent systems, in which the traditional centralized pre-programmed control is replaced by a distributed functioning where the interactions between such individuals lead to the emergence of "intelligent" global behavior, unknown to them [13]. Examples of swarm intelligence include ant colonies, bird flocking, fish shoaling and bacterial growth [8]. The swarm principles were also used to design optimization algorithms, namely the Ant Colony Optimization (ACO) and the Particle Swarm Optimization (PSO).

Translating these ideas to the manufacturing world, manufacturing systems can be seen as a community of autonomous and cooperative entities, the holons in the HMS paradigm, each one regulated by a small number of simple rules and representing a manufacturing component, such as a robot, a conveyor, a pallet or an order. The degree of complexity of the behavior of each entity is strongly dependent of the embodied intelligence and learning skills. Very complex and adaptive systems can emerge from the interaction between the individual entities, as illustrated in Figure 1.

The emergent behavior considers two different levels: the macro level considering the system as a whole and the micro level considering the system from the point of view of the local components. The achieved behavior results from the capability of individual entities to change dynamically and autonomously their properties, coordinated towards a unique goal to evolve. In fact, even if all individuals perform their tasks, the sum of their activities could be the chaos (disorder) if they are not coordinated according to a common goal [14].

In manufacturing, the coordination of these systems, for example for the task allocation, is usually related to the regulation of expectations of entities presenting conflict of interests: some entities (usually products or orders) have operations to be executed and others (usually resources) have skills to execute them. Several algorithms can be used for this purpose, namely those based on the Contract Net Protocol (CNP) [15], those based on the markets laws [16] and those based on the attraction fields concept [17].

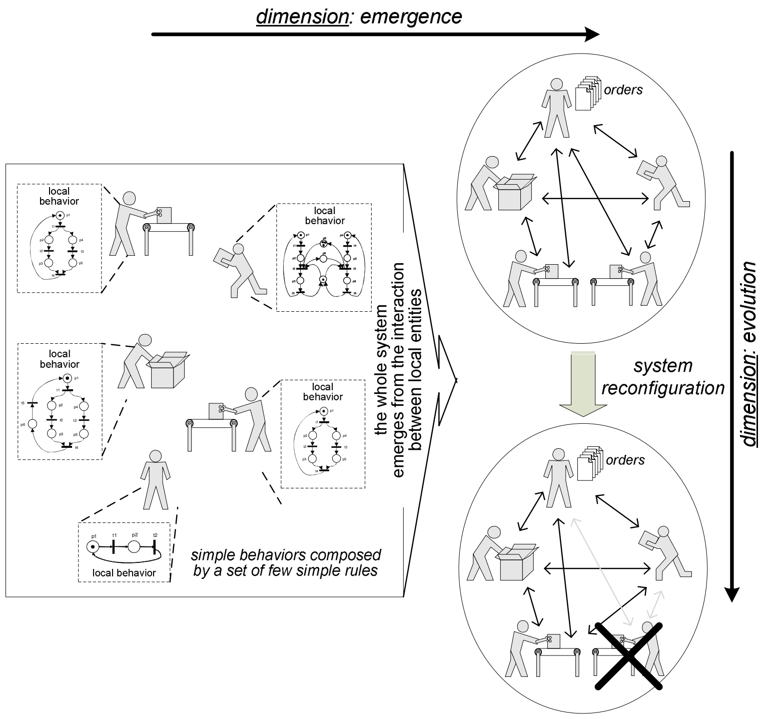


Fig. 1. Bi-Dimensional Approach to Complex Evolvable Systems

Systems exhibiting these features operates in a very flexible and robust way [13]: *flexible*, since it allows the adaptation to changing environments by adding, removing or modifying the entities on the fly, i.e. without the need to stop, re-program and re-initialize the other components, and *robust*, since the society of entities has the ability to work even if some individuals may fail to perform their tasks.

After analyzing how complex systems emerge from the interactions between simple individuals, illustrated by the emergence dimension of Fig.1, the raised question is related to how the system can self-organize and evolve to adapt quickly and efficiently to the environment volatility. This question is illustrated as the evolution dimension also represented in Fig. 1. The holonic and multi-agent applications address the first dimension but rarely consider the second one.

4 Self-adaptation and Self-organization of Complex Systems

Ideally, re-configuration should be done on the fly, maintaining unchanged the behavior of the entire system which should continue to run smoothly after the change. The need for re-configuration and evolution can appear in several manufacturing situations, from the virtual organization level, where reconfiguration of partners is frequent, to the shop floor level, for example in the re-configuration of the part transportation flow, passing by the machine level, for example by changing the set of grippers aggregated to a robot.

In biological systems there are two different approaches for the adaptation to the dynamic evolution of the environment [18]: evolutionary systems and self-organization. The next sections discuss the concepts of evolution and self-organization to achieve adaptive and evolvable systems.

4.1 Evolutionary Theory

The evolutionary approach derives from the theory of evolution introduced by Charles Darwin 150 years ago [19]. Darwin believed that species change over a long period of time, evolving to suit their environment. In the evolution process, the selection is natural in the sense that is purely spontaneous without a pre-defined plan. Although being chaotic and unpredictable, evolution moves preferentially in the direction of increasing a fitness objective, which depends of the system context and strategy: e.g. the objective can be to reduce the thermo-dynamical energy or to increase the system's productivity. Darwin stated that the species that will survive to evolution and changes in the environment are not the strongest or the most intelligent, but those that are more responsive to change.

An advance in the evolutionary theory is the theory of punctuated equilibrium. In opposite to the theory of evolution idealized by Darwin, where the evolution is a slow, continuous process without sudden jumps, the evolution in punctuated equilibrium tends to be characterized by long periods where nothing changed, "punctuated" by episodes of very fast development of new forms.

Translating these theories to the manufacturing world, the companies better prepared to survive in the current competitive markets are those that better respond to emergent and volatile environments, by adapting dynamically their behavior [5]. The complex manufacturing systems should evolve continuously or punctually, driven by external stimulus that force its adaptation and re-organization. The distributed entities are subject to the application of evolutionary techniques by selecting gradually a better system. Well known examples of this approach are the neural networks and genetic algorithms.

4.2 Self-organization

In biological systems, self-organization plays an important role to achieve system's adaptation to the dynamic evolution of the environment. Self-organization is basically a process of evolution where the effect of the environment is minimal, i.e. where the development of new, complex structures takes place primarily through the system itself, being normally triggered by internal variation processes, which are usually called "fluctuations" or "forces".

Self-organization is not a new concept, being applied in different domains such as economics, computing and robotics. Several distinct definitions, but not necessarily contradictory, are found in the literature (e.g. see [20] and the references therein). A possible definition is to consider self-organization as the ability of an entity/system to adapt dynamically its behavior to external conditions by re-organizing its structure through the modification of the relationships among entities without external intervention and driven by local and global forces. The integration of self-organization capabilities may require using distributed architectures [20]. In fact, autonomous systems, as our brain, have to constantly optimize their behavior, involving the combination of non-linear and dynamic processes. These characteristics imply the management and control of behavioral complexity as well. The self-organizing behavior of each entity is based on an adaptive mechanism that dynamically interprets and responds to perturbations [20].

Evolution and self-organization can be confusing concepts. Some authors state that natural selection must be complemented by self-organization in order to explain evolution [21]. In the Darwin's theory of evolution, the evolution is the result of the selection by the environment, acting on a population of organisms competing for resources, i.e. encompassing external forces, while in self-organization, the evolution is purely due to the internal configuration without any external forces or pressures. Also, the difference between self-organization and emergence should be stated: they both lead to systems that evolve over time and can not be directly controlled from the exterior, but while emergent systems consist of a set of individuals that collaborate to exhibit a higher behavior, the self-organized systems exhibit a goal-directed behavior. This emergence/self-organization relationship can be expressed in the bi-dimensional approach to complex evolvable systems illustrated in Fig. 1.

Translating the self-organization concept found in nature to the manufacturing domain, the network of entities that represents the control system is established by the entities themselves. The driving forces guide the re-organization process according to the environment conditions and to the control properties of the distributed entities. Several self-organization mechanisms can be found [22]: foraging, nest building, molding and aggregation, morphogenesis, web weaving, brood sorting, flocking and quorum. Each one of these mechanisms presents different driven forces to support the evolution.

The application of self-organization associated to emergent behaviors allows achieving dynamic evolution and reconfiguration. Namely, it allows the dynamic self-configuration (i.e. adaptation to changing conditions by changing their own configuration permitting the addition/removal of resources on the fly and without service disruption), self-optimization (i.e. tuning itself in a pro-active way to respond to environmental stimuli) and self-healing (i.e. capacity to diagnose deviations from normal conditions and take proactive actions to normalize them and avoid service disruptions) [23].

Stigmergy is a form of self-organization, involving an indirect coordination between entities, where the trace left in the environment stimulates the execution of a subsequent action, by the same or different entity. As an example, ants exchange information by depositing pheromones on their way back to the nest when they have found food. This form of self-organization produces complex, intelligent and efficient collaboration, without the need for planning, control or even communication between the entities.

The embodied intelligence, a concept used in the Artificial Life field, may play another important role in the design of these systems. Embodied intelligence suggests that intelligence requires a body to interact with [24], with the intelligent behavior emerging from the interaction of brain, body and environment. The key issue is to define powerful intelligence mechanisms, not only including static intelligence mechanisms but also learning capabilities, that enable the system to behave better in the future as the result of its experience and knowledge. The degree of efficiency of the self-organization capability, and consequently the improvement of the entity's performance to dynamically evolve in case of emergency, is strongly dependent on how the learning mechanisms are implemented.

4.3 Stability in the Evolution Process

In dynamic and complex systems, in which emergence and evolution play key roles, besides to consider mechanisms to identify the reconfiguration and evolution opportunities, it is important to maintain the system in a stable and predictable state. In fact, during the evolution process some instability and unpredictability can appear as the result of not properly synchronized evolution processes. The concept of stability is concerned to the condition in which a slight disturbance or modification in the system does not produce a significant disrupting effect on that system.

In the complex systems described in the paper, each individual has partial knowledge, i.e. none of them has a global view of the system, introducing uncertainty in the system. Being the entropy a measure of uncertainty of information or ignorance, these systems are normally associated to disorder (chaos). Using mechanisms that combine efficiently the individual knowledge hosted by distributed entities, it is possible to reduce the entropy of the system, becoming the system complex, organized and ordered.

For this purpose, regulation mechanisms are crucial to emerge from chaos to order during the evolution process, achieving stability and avoiding the increase of entropy and consequently the chaotic or instable states. A self-organizing system which decreases its entropy must necessarily, in analogy to the 2nd law of thermodynamics, dissipate such entropy to its surroundings.

5 The ADACOR Example

ADACOR [5] is a holonic control architecture that uses concepts inherited from biology, namely swarm intelligence and self-organization to achieve an adaptive production control approach that addresses the system re-configurability and evolution, especially when operating in emergent environments.

ADACOR architecture is built upon a community of autonomous and cooperative holons, each one representing manufacturing entities, e.g. robots, pallets and orders. In analogy with insect colonies, where an individual usually does not perform all tasks, but rather specializes in a set of tasks [13], ADACOR architecture identifies four manufacturing holon classes, each one possessing proper roles, objectives and behaviors [5]: product (PH), task (TH), operational (OH) and supervisor (SH). The product holons represent the products available in the factory catalogue, the task holons represent the production orders launched to the shop floor to execute the requested products and the operational holons represent the physical resources available at shop floor. Supervisor holons provide co-ordination and optimization services to the group of holons under their supervision, and thus introducing hierarchy in a decentralized system.

The modularity provided by ADACOR is similar to that exhibited by the Lego™ concept: grouping elementary and inter-connectable entities in a particular way it is possible to build bigger and more complex systems. The systems' re-configurability or evolution is achieved by the dynamic re-aggregation of the elementary components or systems. Being the ADACOR holons pluggable (i.e. without the need to re-initialize and re-program the system when a holon is added to the system), it offers enormous flexibility and re-configurability to support emergent behavior on the fly.

The system self-organization is only achieved if the distributed entities have stimulus to drive their local self-organization capabilities. The self-organization is regulated by the autonomy factor, which fixes the level of autonomy of each holon, and evolves dynamically in order to adapt the holon behavior to the changes in the environment where it is placed. The evolution is governed by a decision mechanism where learning mechanisms play a crucial role to detect evolution opportunities and ways to evolve. These two local driving forces (i.e. autonomy and learning) allow the dynamic self-adaptation of the holon, contributing for the re-configuration of the system as a whole.

The global driving force used in ADACOR to support the system's self-organization is a pheromone-like spreading mechanism. The propagation of the need for re-organization is done by depositing a certain quantity of pheromone, proportional to the forecasted impact of the disturbance, in the neighbor supervisor holons (i.e. using the stimergy concept). The holons associated to each supervisor holon sense the pheromone, and trigger a self-adaptation of their behavior (e.g. increasing its autonomy) and propagate the pheromone to other neighbor holons. The intensity of the pheromone odor becomes smaller as far it is from the epicenter of the evolution trigger, i.e. similar to a field gradient.

Fig. 2 illustrates how an adaptive production control can be achieved by using the described self-organization mechanisms. Briefly, in stationary operation, when the objective is the optimization, the holons are organized in a hierarchical structure, with the presence of supervisor holons, acting as coordinating entities. Supervisor holons elaborate, periodically, optimized schedules that are proposed to the operational holons, which as enough autonomy to follow or not these suggestions. When an unexpected disturbance is detected, the system is forced to evolve to a heterarchical structure, operating without the presence of coordination levels, with each holon increasing its autonomy and assuming the complete control of its own activity. This re-organization is supported by the self-adaptation of each holon and the self-organization of the system propagated using the pheromone based technique. In this stage, the scheduling is achieved in a distributed manner, resulting solely from the interaction between task and operational holons.

After to recover from the disturbance, the system evolves to a new control structure, which can be the previous one or a new one, according to the learning mechanisms embedded in each holon.

The described approach was implemented using the JADE (Java Agent Development Framework) multi-agent framework and applied to a flexible manufacturing system [25]. The achieved experimental results allowed to conclude the applicability and benefits of the described adaptive production control approach and also the need to apply more powerful reconfigurable and self-organizing mechanisms. In fact, the evolvable mechanisms should be more continuous, unpredictable and self-regulated based on more powerful self-organization techniques combined with the embodiment of more learning capabilities in the distributed entities.

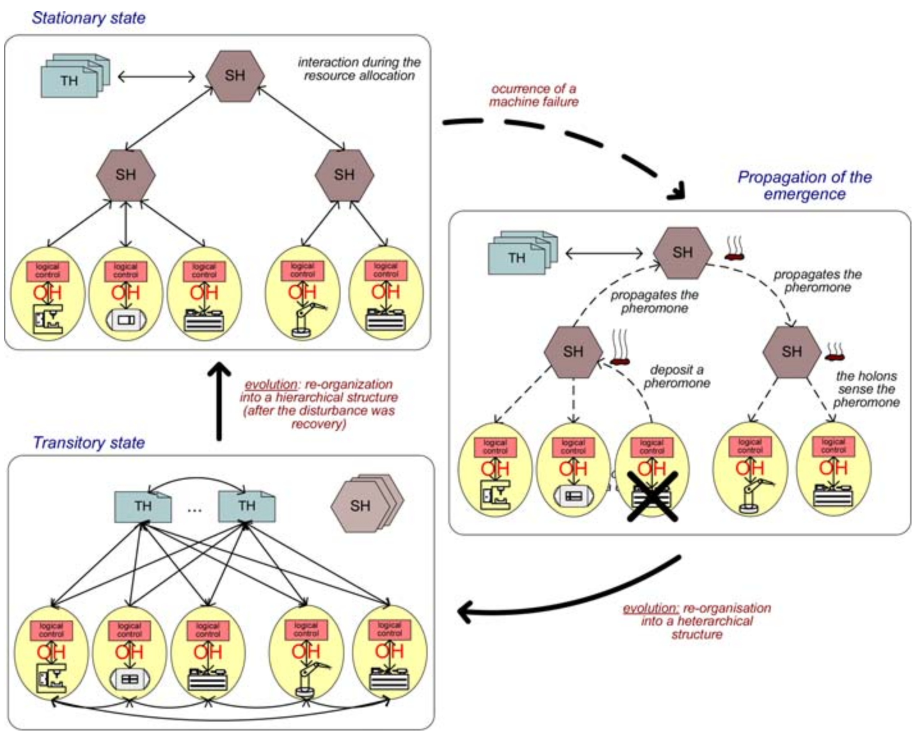

Stationary state
interaction during the resource allocation
TH
SH
logical control
OH
ocurrence of a machine failure
Propagation of the emergence
propagates the pheromone
deposit a pheromone
the holons sense the pheromone
evolution: re-organization into a hierarchical structure (after the disturbance was recovery)
Transitory state
evolution: re-organisation into a heterarchical structure

Fig. 2. Adaptive Production Control based on a Self-organization Mechanism

6 Conclusions

Aiming to address the emergent requirements imposed to the manufacturing domain, a current challenge to design re-configurable and responsiveness manufacturing systems is to consider the holonic design combined with the potential offered by biological inspired techniques, namely swarm intelligence and self-organization. Understanding how in nature the complex things are performed in a simple and effective way, allows us to copy and develop complex and powerful adaptive and evolvable systems. Additionally, these concepts can be combined with insights from emergent theories from computer science, namely the evolutionary computing and artificial life.

The paper makes an incursion in these biological inspired techniques, namely the self-organization concept, usually associated with complex and non-linear phenomena. It was pointed out that the interaction between simple individuals may create intricate and unpredictable patterns (chaos), which can evolve until they reach a stable configuration (order), i.e. emerging from chaos to order and being the basis for self-evolvable systems.

The ADACOR holonic approach was used to illustrate the benefits resulted from the application of some biological inspired theories, namely emergence and self-organization, to achieve an adaptive production control that evolves from centralized structures when the objective is the optimization, to more flat structures in presence of unexpected scenarios.

The further work is related to the design of more powerful and evolvable mechanisms supporting the operation of non-linear and dynamic processes, as manufacturing systems are, exhibiting desired features of modularity, reconfigurability, responsiveness and robustness.

References

1. Wooldridge, M.: An Introduction to Multi-Agent Systems. John Wiley & Sons, Chichester (2002)
2. Minsky, M.: The Society of Mind. Heinemann (1985)
3. Okino, N.: Bionic Manufacturing System. In: Peklenik, J. (ed.) CIRP Flexible Manufacturing Systems Past-Present-Future, pp. 73–95 (1993)
4. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers in Industry 37, 255–274 (1998)
5. Leitão, P., Restivo, F.: ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control. Computers in Industry 57(2), 121–130 (2006)
6. Chirn, J.-L., McFarlane, D.: A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture. In: Proceedings of the International Workshop on HoloMAS, pp. 219–223 (2000)
7. Schild, K., Bussmann, S.: Self-organization in Manufacturing Operations. Communications of the ACM 50(12), 74–79 (2007)
8. Miller, P.: The Genius of Swarms. National Geographic (July 2007)
9. Valckenaers, P., Hadeli, K.M., Brussel, H., Bochmann, O.: Stigmergy in Holonic Manufacturing Systems. Journal of Integrated Computer-Aided Engineering 9(3), 281–289 (2002)
10. Koestler, A.: The Ghost in the Machine. Arkana Books, London (1969)
11. Marik, V., McFarlane, D.: Industrial Adoption of Agent-based Technologies. IEEE Intelligent Systems 20(1), 27–35 (2005)
12. Holland, J.: Emergence: from Chaos to Order. Oxford University Press, Oxford (1998)
13. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: from Natural to Artificial Systems. Oxford University Press, Oxford (1999)
14. Thamarajah, A.: A Self-organizing Model for Scheduling Distributed Autonomous Manufacturing Systems. Cybernetics Systems 29(5), 461–480 (1998)
15. Smith, R.: Contract Net Protocol: High-Level Communication and Control in a Distributed Solver. IEEE Transactions on Computers C-29(12), 1104–1113 (1980)
16. Markus, A., Vancza, T.K., Monostori, L.: A Market Approach to Holonic Manufacturing. Annals of CIRP 45, 433–436 (1996)
17. Vaario, J., Ueda, K.: Biological Concept of Self-organization for Dynamic Shop Floor Configuration. In: Proc. of Advanced Product Management Systems, pp. 55–66 (1997)
18. Vaario, J., Ueda, K.: Self-Organisation in Manufacturing Systems. In: Japan-USA Symposium on Flexible Automation, Boston, US, pp. 1481–1484 (1996)
19. Darwin, C.: The Origin of Species. Signet Classic (2003)
20. Bousbia, S., Trentesaux, D.: Self-organization in Distributed Manufacturing Control: State-of-the-art and Future Trends. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 5 (2002)

21. Kauffman, S.: The Origins of Order: Self Organization and Selection in Evolution. Oxford University Press, New York (1993)
22. Manei, M., Menezes, R., Tolksdorf, R., Zambonelli, F.: Case Studies for Self-organization in Computer Science. Journal of Systems Architecture 52, 443–460 (2006)
23. Leitão, P.: A Bio-Inspired Solution for Manufacturing Control Systems. In: Azevedo, A. (ed.) IFIP International Federation for Information Processing, Innovation in Manufacturing Networks, pp. 303–314. Springer, Heidelberg (2008)
24. Pfeifer, R., Scheier, C.: Understanding Intelligence. MIT, Cambridge (2001)
25. Leitão, P., Restivo, F.: Implementation of a Holonic Control System in a Flexible Manufacturing System. IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews 38(5), 699–709 (2008)

Service-Oriented Agents for Collaborative Industrial Automation and Production Systems

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

[1] Faculty of Engineering – University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
{marco.mendes,fjr}@fe.up.pt
[2] Polytechnic Institute of Bragança, Quinta S^{ta} Apolónia, Apartado 134, 5301-857 Bragança, Portugal
pleitao@ipb.pt
[3] Schneider Electric Automation GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany
armando.colombo@de.schneider-electric.com

Abstract. Service-oriented Multi-Agent Systems (SoMAS) is an approach to combine the fundamental characteristics of service-oriented and multi-agent methods into a new platform for industrial automation. Several research works already targeted the connection of these technologies, presenting different perspectives in how and why to join them. This research focuses on available efforts and solutions in the area of SoMAS and explains the idea behind the service-oriented agents in industrial automation. A SoMAS system is mainly composed by shared resources in form of services and their providing/requesting agents. The paper also discusses the required engineering aspects of these systems, from the internal anatomy to the interaction patterns. Parameters of flexibility, reconfiguration, autonomy and reduced development efforts were considered and they should be the trademark of SoMAS. Aiming to illustrate the proposed approach, an example of service-oriented automation agents is given.

Keywords: Multi-agent Systems, Service-oriented Architectures, Industrial Automation.

1 Introduction

Current industrial automation and production systems are complex, heterogeneous in nature and require enormous development and maintenance efforts. The growing needs for more flexible and reconfigurable production systems has led to the development of new paradigms. One promising approach is to have a conglomerate of autonomous, reusable and co-operating entities (objects) that, under a functional point of view, are capable of dynamically interacting with each other to achieve both local and global objectives. When they are considered within a cross-layer infrastructure, like a manufacturing enterprise [1], some of them are embedded into automation devices and others are available in computing devices for more complex tasks. Attending to the nature and requirements of such systems, several software methodologies are being considered: the open standard of IEC 61499 to build functional block-based automation control systems, the more experimental Multi-Agent Systems (MAS) and more recently Service-oriented Architectures (SoA). Although one of the common objectives of these methodologies is to provide a sort of interoperability and flexibility over distributed automation systems, their background and specification is quite dissimilar.

From the research and development background of the authors on agent-based, service-oriented and component-based systems for automation (see for example the publication [2]), it was observed that the existing service-oriented entities that were specified, assimilate agents in terms of autonomy over a particular problem/resource and their "social" behavior. This observation motivated a research in the field of integrating multi-agent and service-oriented systems. This suggestion is not new since services are already part of agents' specification, such as in FIPA (Foundation for Intelligent Physical Agents) [3], and agents are also present in standard documents of SoA methodologies [4]. Thereof, the under-considered elements (services in MAS and agents in SoA) are vaguely defined and have a more passive and customized role. Both agents and services have a very active function in the developed work for industrial automation, intersecting them to form the so-called Service-oriented Multi-Agent Systems (SoMAS). Most research dealing with the convergence are more focused on interoperability problems, semantic descriptions and simplification of problems with compositions, but the real nature, anatomy, behavior, interactions and autonomy of service-oriented agents is still less considered.

The concepts and studies introduced in this paper represents the authors' experience in converging multi-agent and service-oriented systems in industrial automation and production systems, resulting in the form of service-oriented multi-agent systems. Following the introduction, section 2 reviews the application of multi-agent and service-oriented systems in industrial automation and discusses the integration of both paradigms. Section 3 introduces the adopted approach for service-oriented multi-agent systems and illustrates it with a simple example. Finally, section 4 resumes the conclusions and defines future work.

2 Background Information and Review of Literature

Service-oriented Architectures is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. This definition is according to the Reference Model for Service Oriented Architecture [4], which indicates an abstract framework for understanding significant entities and relationships between them within a service-oriented environment, and for the development of consistent standards or specifications supporting that environment. SoA is a means of organizing solutions that promotes reuse, growth and interoperability. Visibility, interaction, and effect are key concepts for describing the SoA paradigm. Note that although SoA is usually implemented using Web Services (WS), services can be made visible, support interaction, and generate effects through other implementation strategies. In a similar way, agent systems are frequently developed to be compliant to the IEEE FIPA standards. FIPA Abstract Architecture [3] includes a specification that defines architectural elements and their relationships.

2.1 Overview of Agent and Service Systems in Industrial Automation

Agent systems were introduced in automation and manufacturing when there was the vision of having distributable software components to resolve local tasks, comparable to the heterogeneous nature of the physical domain. Existing models of production, such as Flexible Manufacturing Systems (FMS) [5], Reconfigurable Manufacturing Systems (RMS) [6] and Collaborative Manufacturing Management (CMM) [7], are a basis for the application of multi-agent systems to solve and optimize the problems of production, due to the increase of introduced complexity [8]. Among the several publications reporting the application of MAS in automation and production systems, the reader can consult the reference [9] to retrieve more information on this topic.

Service-oriented systems are well known in the field of business and electronic commerce but in industrial automation systems (especially concerning distributed devices) it is a relatively new research area with promising results. The root of SoA fits well with collaborative automation, in the sense of autonomous, reusable and loosely-coupled distributed components. In automation domain, the vision of using SoA is to support the life-cycle needs in the context of agile and flexible manufacturing, addressing distributed, modular and reconfigurable automation systems which behavior is regulated by the coordination of services. The SIRENA Project [10] has contributed to strength the SoA-based automation by providing Web services at device level through the extension of the SoA paradigm into the realm of low-level embedded devices, such as sensors and actuators. Since then, significant research is going on, covering the engineering of such systems, including the process modeling, semantic description and collaboration. Research projects such as SOCRADES [11] and SODA [12] are contributing to develop extensions in service-oriented automation systems.

The integration of SoA with MAS in industrial automation and concretely its benefits are still unknown, but had started to be discussed. Being both currently the most promising concepts in this matter, the suggestion is that there are substantial benefits in converging both paradigms and technologies [13]. Other conclusions can be obtained from the application in other fields, as described in the next subsection.

2.2 Integration of Agents and Services

MAS and SoA are two paradigms that are based on separated models, but which also share common background requirements, features and applications.

The concept of service is frequently presented in publications about agent systems. In [14], services are used to provide a generic scheme to describe interactions between two agents. The authors of [15] propose an extension of the Web services-agent integration toolkit WS2JADE for Web services management with FIPA compliant multi-agent systems. The AgentWeb Gateway is an initiative for dynamic and seamless interoperation of multi-agent systems and Web services [16]. In [17], an agent-based service-oriented system architecture for manufacturing enterprise collaboration is presented. The authors of [18] explore how much the most common agent-oriented methodologies are also oriented to the development of services, comparing different agent-oriented methodologies under the point of view of the development of services. Integration of services is also discussed in [19], but rather presents available solutions in a more technological point of view. Agents have being widely studied in the way of creating compositions and orchestrations of services that can be understood as a simplification of services in a single one [20]. The power of reasoning provided by agents and the facility of semantic descriptions that are available in services is identified as a strong linkage to prove the usefulness of combining both solutions. As an association, the composition of (Web) services is similar to the planning problem that has been investigated extensively in artificial intelligence [21], including multi-agent systems.

In the same way that the word *service* is known in the "agent world", the terminology of *agent* is not new to SoA (see [4]). The main issue is that both paradigms have a distinguish evolution and traditional application fields; thereof they are more focused on their specific questions. According to [22], agent-based services are different from (Web) services, because they can support forming big services automatically and dynamically by the way of liberal coalition themselves in that each participant is autonomous.

The convergence of MAS and SoA can be discussed in different ways. From one side, the idea is to create interoperability mechanisms between the "agent world" and the "service world", using specific gateways/proxies that normally translate the semantic from one side to the other by the means of protocol translation (see [15] and [16]). Another possibility is to encapsulate single agents as services (agent-based service) and thus having a direct access to other services (see [17] and [23]). A completely different approach is the concept of service-oriented agents, discussed in this paper, that not only share services as their major form of communication, but also complement their own goals with external provided services (see [24]). Figure 1 depictures graphically the three approaches.

Some researchers discuss subjects such as intelligent services [25] or agent as services [23], but from our side a pertinent question is: should the service be intelligent or its provider/requester entity? A clear separation must be handled between the shared resource/action (service) and the entity that is responsible of the share. A simple example is a nurse offering a healthcare service to a patient. Who is intelligent? The healthcare service, the nurse or the patient? The answer would be the nurse and the patient, since the healthcare service is just a resource that is offered. This also demonstrates a strong linkage between services and the entities that provide/request them. Intelligent providers can decide over different steps of the service and requesters may able to negotiate previously before accepting the service, as an example. Agents may be the service's providers and requesters, since their definition fits well with this concept.

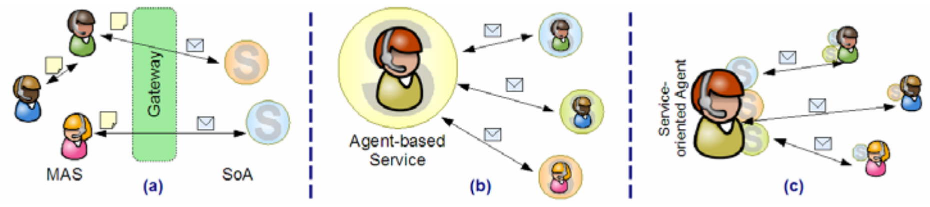


Fig. 1. Three commonly used approaches for integrating SoA and MAS: (a) interoperability gateway between MAS and SoA, (b) SoA encapsulating agents, (c) SoMAS

The challenge is thus the enrichment of SoA with some of the MAS characteristics (or vice-versa), to provide a more dynamic and flexible approach on how services are exposed and consumed, and how collaboration can be done using service-orientation. Cooperation is necessary and essential for multi-agent systems. In most of applications, especially in Internet-based distributed systems, service's request and offer are the main cooperation manners among agents, where agent either requests to get services or offers services for other agents. Such a system can be taken as service-oriented system [26]. The agent's characteristics of pro-action and autonomy, and agents' ability to negotiate commitments and deal with exceptions, are needed for the anticipated applications of service-oriented computing [23]. Therefore, agents can be used to build high-level models with flexible interaction patterns, while Web services are more suitable for solving interoperability problems among heterogeneous applications across enterprises [17].

However, issues regarding how they may be competing or complementary technologies remain open. Because of that, research involving agents and Web services is mainly focused on building improved semantics, communication languages and interaction protocols [27]. This represents one research direction where interoperability and the meaning of things are key issues to drive the use of fixed protocols into new directions (see [28]). The other one is reducing complexity: the concept of agents used in the Web service world is related to solve complex problems on the Web or as intermediaries' entities that make a complex organization externally accessible, and related to the way of simplifying the interface [20]. Reduced research has been seen in terms of the nature of service-oriented agents by the means of the necessity of requesting services, the conditions to provide services and their autonomy over a specific part of the overall (automation) problem.

3 An Approach for Service-Oriented Multi-agent Systems

The main requirement of modern industrial automation and production systems is to provide enough flexibility for automatic reconfiguration, able to respond promptly to changes in the environment and to client demands. To support the service-orientation of the systems that the authors were exploring, software components were specified and developed (see [2]). These service providing/requesting software components generally are used to represent mechatronic devices and enabled with a certain degree of autonomy to control the device, designated as Smart Mechatronic Components (SMeC). The interaction (or collaboration to reach global production objectives) is done via the access to the provided services. This work is reported in [2], where the reader can obtain more information.

The autonomy, mediation/representation and ability for collaborative processes that the previously explained components manifest are similar to what normal agents should have. An agent, according to [29] is a computer system situated in some environment, and that is capable of flexible autonomous actions in its environment in order to meet its design objectives. The flexibility of an agent is characterized by its responsiveness, pro-activeness and social behavior. The intelligence is still a matter of subjective appreciation, but according to the previous definition, the cited SMeCs exhibit the main features associated to an (intelligent) agent.

From this observation, a motivation rose in exploring the area of SoA and MAS paradigms to extract useful features and a new form of building automation systems. The adopted solution comes in form of a Service-oriented Multi-Agent System (SoMAS) that combines elements from both models, but also maintains a certain degree of retro-compatibility to its originators. The subsequent sections discuss the adopted approach based on the research work and the existing requirements to build service-oriented agents for industrial automation.

3.1 The SoMAS Approach

The proposed approach is characterized by using a set of distributed autonomous and cooperative entities, the agents, using the SoA principles. The development of such SoMAS, presenting differences in relation to the traditional MAS, requires an engineering development framework that supports the new functionalities. More specifically, it describes a model for the development of a MAS oriented by the offer and request of services, in order to fulfill industrial and production systems goals. The approach is different from traditional MAS in the form of:

– *Agents are service-oriented*: individual goals of agents may be complemented by services provided by other agents. Internal conditions of agents can be offered as services for others to be requested.
– *Features inherited from service-orientation*: a main feature is the re-usability of services of an agent that can be used and applied not only for a single type of a problem for what it was developed, but also be re-used for different tasks.

The model provides the general guidelines to specify SoMAS for resolving problems and achieve goals in industrial automation and productions. Figure 2 shows the conceptualization based on the different key concepts and an overview of the system's concept. A brief description about each key concept follows:

– *Domain - Industrial Automation and Production*: the domain of the model is applied for industrial automation and production, made of software and hardware components.

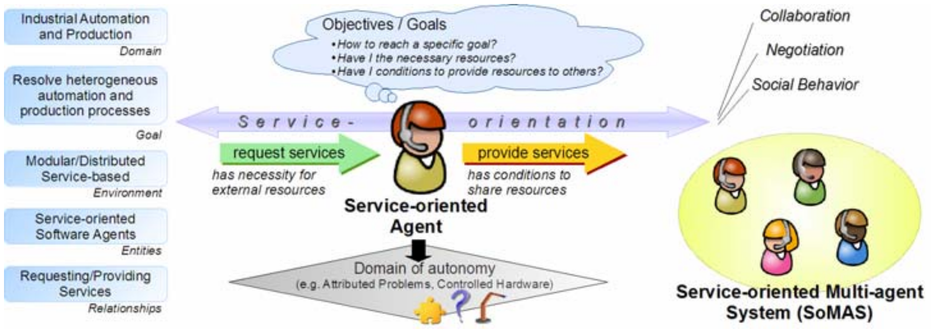

Industrial Automation and Production
Domain
Resolve heterogeneous automation and production processes
Goal
Modular/Distributed Service-based
Environment
Service-oriented Software Agents
Entities
Requesting/Providing Services
Relationships
Objectives / Goals
•How to reach a specific goal?
•Have I the necessary resources?
•Have I conditions to provide resources to others?
Service- orientation
request services
has necessity for external resources
provide services
has conditions to share resources
Service-oriented Agent
Domain of autonomy
(e.g. Attributed Problems, Controlled Hardware)
Collaboration
Negotiation
Social Behavior
Service-oriented Multi-agent System (SoMAS)

Fig. 2. Model and conceptualization of SoMAS

– *Goal - Resolve heterogeneous automation and production processes*: since modern and flexible systems are made of and involve heterogeneous and concurrent activities, the model is targeting the management and resolution of processes for the automation of devices and also the production of goods. Processes may comprehend simple statically descriptions of work-plans to fulfill a given goal or dynamically constructed collaborations for a more flexible approach.
– *Environment - Modular/Distributed, Service-based*: in the system there are software and hardware components. Software components are generally agents that have specified tasks and also may represent/control hardware components. Resources and functions of agents and other software components are exposed as services.
– *Entities - Service-oriented Software Agents*: a software agent is an entity that may act for a user or device and has the ability to achieve their own goals. Several additional concepts include the degree of autonomy of agents and also their learning and intelligence capabilities. A multi-agent system is used when there are complex objectives made of separable individual problems that can be resolved by single agents and their collaboration, reducing the complexity of the overall problem.
– *Relationships - Requesting/Providing Services*: problems and sub-problems are solved by agents that should collaborate. The collaboration is achieved by providing services by agents (when they have the conditions) and their requesting (when agents have needs).

Since this specification represents only a general model, the communication and the development technology is open. This means that any suitable technology can be used to develop the concepts presented by the model. For example, Web services technologies can be adapted to the resource sharing via services and C++ or Java languages can be used to develop the agents.

3.2 Engineering Aspects

Agents will not "work" by their own unless some basic specifications for the engineering related to the domain are considered. Several additional topics are related to the engineering of systems that are specified and developed according to the proposed concept. From the other side and since the model is technology independent, the development of consequent service-oriented multi-agent systems can be done using the current available agent development frameworks and service-oriented systems.

The FIPA Abstract Architecture [3] explicitly avoids the issues related to the internal structure of an agent. It also largely defers details of agent services to more concrete architecture documents. The same openness is also exhibited by the proposed approach, thus each agent may be implemented internally independently and differently (i.e. using different technologies, e.g. different decision techniques and different programming languages). The only requirement is that it should share its functions as services and obey to the protocols of communication and processes that were established. This flexibility may contribute for the development of customized agents for diversified tasks. A suggestion for the internal specification of these service-oriented agents is the adoption of a modular approach as the one described in [30]. It introduces an anatomical-like structure for the development of functional and reusable modules of a component/agent, part of a service-oriented automation system.

One of the most important issues is the interaction among entities. A specification for interaction patterns for service-oriented entities was elaborated and documented in [31], similar to the interaction approach of agent systems. The proposed solution defines a set of interaction phases for the use of services: Discovery, Negotiation, Operational and Termination. Each one of these phases is managed by specific ports of the service. For example, if a requester would like to access a transportation service provided by some agent, it would first enter in the discovery phase and then proceed to the negotiation phase, accessing a specific negotiation port of the discovered transportation port. After successful negotiation, the operation of the services is activated by the operational port. Termination routines can then be accessed via the termination port.

Besides the interaction, the following engineering aspects have also to be considered in the engineering framework:

– *Description of knowledge and semantics*, i.e. agents may understand what is the information associated to services and other elements.
– *Centralization/decentralization of entities (agents) and tasks*: due to the flexibility of the model, implemented systems may adopt a more centralized approach (e.g. master/slave agents) or a completely decentralized distribution (in a more collaborative fashion).
– *Description of processes* that describe the behavior of agents and usage of services can be used as a complement for coordination of activities.
– Services may be *aggregated/associated*, simplifying the external access to resources.
– A way to *address and discover* services and agents.
– *Autonomy and pro-activity* of agents should be considered in the sense of not only providing services, but also in the ability of requesting them and having "autonomy" of decision and control over some part of the problem.
– *Design and agent development tools,* i.e. providing an environment and framework for easy development and monitoring of agents and services.
– *Security aspects.*

Depending on the type of implementation, other requirements for engineering may also be necessary, besides the ones previously referred.

3.3 Example of a Service-Oriented Agent

There are several examples and applications that can be given based on previous explained concept. A traditional manufacturing scenario includes several components that can be distinguished: processing machines, transport systems, warehouses and the products itself. Considering a multi-agent system applied to this scenario, software agents would represent the several elements in the system and act over them.

A concrete example agent is applied as a mediator of a virtual unidirectional conveyor to transport pallets from one port to another (see Figure 3). As such, the agent has several functional modules: Communication and Device Interface. These modules are part of the agent's anatomy and permit the agent to communicate via service-orientation and access the physical conveyor, respectively. In its core the agent provides a service (*Transfer* service) because it has the conditions to do so. But to proceed to a specific transfer of pallets, that is its goal, it also needs to transfer the

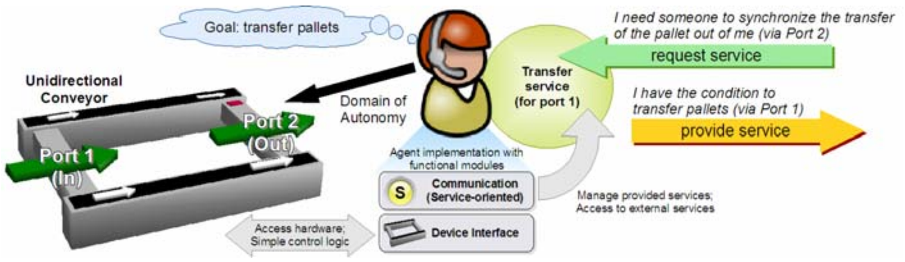

Goal: transfer pallets
Unidirectional Conveyor
Port 1 (In)
Port 2 (Out)
Domain of Autonomy
Transfer service (for port 1)
I need someone to synchronize the transfer of the pallet out of me (via Port 2)
request service
I have the condition to transfer pallets (via Port 1)
provide service
Agent implementation with functional modules
Communication (Service-oriented)
Device Interface
Manage provided services; Access to external services
Access hardware; Simple control logic

Fig. 3. Example of a service-oriented agent (mediator of a bidirectional conveyor)

received pallet to a destination. Consequently, it will request a concrete service from a neighbor device that provides the necessary capabilities. This sort of interactions by complementing own objectives with service-orientation, results in a collaboration in benefit of the overall production and building complex nets of automation systems.

The several keywords used previously and that represent the features of SoMAS are explained below using the example of Fig. 3:

– *Modularity and autonomy*: automation and production problems are subdivided and one agent is autonomous in charge of part of it. In the example, the agent is responsible for the management and objectives of the unidirectional conveyor.
– *Interoperability and orientation*: agents are service-oriented and this characteristic strengths the interoperability between different entities in the system. In the example, the agent is able to create its own services, but in order to respect the global goals it must be able to use other services that are in the system. For the full function of the conveyor, services provided by neighbor transport systems should be used by the local agent to transport pallets to the required designations.
– *Transparency and reduced development efforts*: agents can be developed using different methodologies and technologies, but should obey to interaction patterns that are defined and standard to each other. Proposed methodologies such as the internal anatomy for agents [30] and interaction patterns [31] were specified to reduce development efforts and increase functionality.
– *Flexibility and reconfiguration*: system flexibility is introduced by the autonomy and modularity of agents, which can be used for reconfiguration purposes (e.g. local adjustments without stopping the whole system, environmental changes, and modification of production objectives).
– *Integration*: the use of service-oriented principles over multi-agent systems simplifies the enterprise vertical integration, namely into business levels (commonly operating in a service-oriented way).

3.4 Application of Agent and Service Frameworks

The development of a framework based on the present concepts considered several existing implementations. JADE (Java Agent DEvelopment Framework), a widely used framework in the academic area for R&D purposes, is a software framework written in the Java language for the development of FIPA-compliant agents and agent-based systems. Other examples of frameworks for developing agent systems can be referred, such as FIPA-OS, JACK and Mobile-C. One of the main problems of the current agent frameworks is that they have always some sort of centralized management (e.g. containers, agent management system, and directory facilitator) and are not truly distributed and decentralized.

For SoA, it is more difficult to find a complete framework to develop service-oriented systems, being usually only available a set of tools for the development of specific features of the architecture. For example, the SoA for Devices (SOA4D) open toolkit can be used for the development of service-oriented software components in Device Profile for Web Services (DPWS). Therefore, it is only able to create the necessary structures for providing and requesting services, including also the discovery, addressing, etc. One point in favor, besides creating a basis for service-oriented communication, is the possibility of creating real (100%) distributed systems without central management (such as it happens in the current agent frameworks). For example the dynamic discovery permits localization of services without having a central yellow pages server. Other characteristic is the possibility of dynamically configure devices and services using dynamic deployment.

Since the authors were already working on a framework for service-oriented components [32], the previous development approach was also considered for developing service-oriented agents, and to cover some of the previous discussed limitations present in SoA and agent frameworks. Some required features include development on C/C++ languages (targeting multi-platform and specially embedded devices), source code availability and extendable, modular and functional body of agents, and a certain degree of autonomy (not only from the concept of autonomous agent, but also from the point of view of the final software component). Considering the requirements, the existing *Continuum* project was used [32]. The *Continuum* project aims the development and support of the engineering processes in industrial automation systems, considering also the requirements described in this document.

4 Conclusions and Future Work

This paper introduces an approach for the development of distributed intelligent systems that exhibit modularity, flexibility, re-configurability and interoperability features, combining the best features of MAS and SoA paradigms. For this purpose, research was done in the direction of SoMAS, being explained the idea of service-oriented agents. The SoMAS approach introduces main advantages over the traditional MAS and SoA paradigms by combining the best features of MAS and SoA paradigms enhancing the achievement of autonomy, modularity, flexibility and interoperability, and consequent reconfiguration that are required by distributed control systems. Future work is related to enhance the specification of SoMAS and to continue with the development of the framework to be able to specify service-oriented agents for real and virtual automation scenarios. An important question to be resolved is to adopt the compliance with the FIPA specifications.

Acknowledgments

The authors would like to thank the European Commission and the partners of the EU IST FP6 project "Service-Oriented Cross-layer infrastructure for Distributed smart

Embedded devices” (SOCRADES), the EU FP6 "Network of Excellence for Innovative Production Machines and Systems” (I*PROMS), and the EC ICT FP7 project “Cooperating Objects Network of Excellence” (CONET) for their support.

References

1. Deen, S.M.: Agent Based Manufacturing: Advances in the Holonic Approach. Springer, Berlin (2003)
2. Mendes, J.M., Leitão, P., Colombo, A.W., Restivo, F.: Service-oriented control architecture for reconfigurable production systems. In: Proceedings of the 6th IEEE International Conference on Industrial Informatics, pp. 744–749 (2008)
3. FIPA Abstract Architecture Specification. Standard of the Foundation for Intelligent Physical Agents (2002), `http://www.fipa.org/specs/fipa00001`
4. Reference Model for Service Oriented Architecture 1.0. OASIS Standard (October 12, 2006), `http://docs.oasis-open.org/soa-rm/v1.0`
5. Tombak, M., De Meyer, A.: Flexibility and FMS: an empirical analysis. IEEE Transactions on Engineering Management 35(2), 101–107 (1988)
6. Mehrabi, M.G., Ulsoy, A.G., Koren, Y.: Reconfigurable manufacturing systems: Key to future manufacturing. Journal of Intelligent Manufacturing 11(4), 403–419 (2000)
7. Gorbach, G., Nick, R.: Collaborative Manufacturing Management Strategies. White paper, ARC Advisory Group (2002)
8. Tnazefti-Kerkeni, I., Arantes, L., Moalla, M.: An agent-oriented architecture for F.M.S. control/monitoring. In: Proceedings of 2003 IEEE Conference on Control Applications, vol. 2, pp. 1024–1028 (2003)
9. Marik, V., McFarlane, D.: Industrial adoption of agent-based technologies. IEEE Intelligent Systems 20(4), 27–35 (2005)
10. Jammes, F., Smit, H.: Service-oriented architectures for devices - the SIRENA view. In: Proceedings of the 3rd IEEE International Conference on Industrial Informatics, pp. 140–147 (2005)
11. Taisch, M.: The Socrades European project (Service-Orientated Cross-layer InfRAstructure for Distributed Smart Embedded Devices). In: Second World Congress on Engineering Asset Management (EAM) and The Fourth International Conference on Condition Monitoring (2007) (presentation)
12. Mensch, A., Depeisses, F.: SODA Technical Framework Definition (2007), `http://www.soda-itea.org/Documents/Specifications/1176731057.06.html`
13. Ribeiro, L., Barata, J., Mendes, P.: MAS and SOA: Complementary Automation Paradigms. In: IFIP International Federation for Information Processing, vol. 266, pp. 259–268. Springer, Boston (2008)
14. Sesseler, R.: Building agents for service provisioning out of components. In: Proceedings of the fifth international conference on Autonomous agents, pp. 218–219. ACM Press, New York (2001)
15. Nguyen, X.T., Kowalczyk, R.: Enabling agent-based management of Web services with WS2JADE. In: Proceedings of the Fifth International Conference on Quality Software, pp. 407–412 (2005)
16. Shafiq, M.O., Ali, A., Farooq Ahmad, H., Suguri, H.: AgentWeb Gateway - a middleware for dynamic integration of multi agent system and Web services framework. In: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pp. 267–268 (2005)

17. Shen, W., Ghenniwa, H., Li, Y.: Agent-Based Service-Oriented Computing and Applications. In: Proceedings of the 1st International Symposium on Pervasive Computing and Applications, pp. 8–9 (2006)
18. Cabri, G., Leonardi, L., Puviani, M.: Service-Oriented Agent Methodologies. In: Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 24–29 (2007)
19. Greenwood, D., Lyell, M., Mallya, A., Suguri, H.: The IEEE FIPA approach to integrating software agents and web services. In: Proceedings of the 6th international joint conference on autonomous agents and multi-agent systems. ACM, New York (2007)
20. Walton, W.: Agency and the Semantic Web. Oxford University Press, USA (2006)
21. Singh, M., Huhns, M.: Service-Oriented Computing: Semantics, Processes, Agents. John Wiley & Sons Ltd., England (2005)
22. Yang, X., Feng, Z., Xu, G.: The Automatic Formation of Agent Societies: A Service-Driven Approach. In: Proceedings of the International Conference on Advanced Language Processing and Web Information Technology, pp. 556–561 (2008)
23. Huhns, M.N.: Agents as Web services. IEEE Internet Computing 6(4), 93–95 (2002)
24. Paulino, H., Lopes, L.: A service-oriented language for programming mobile agents. In: Proceedings of the fifth International joint Conference on Autonomous Agents and Multi-agent Systems, pp. 1294–1296. ACM Press, New York (2006)
25. Shen, W., Li, Y., Hao, Q., Wang, S., Ghenniwa, H.: Implementing collaborative manufacturing with intelligent Web services. In: Proceedings of the Fifth International Conference on Computer and Information Technology, pp. 1063–1069 (2005)
26. Xinjun, M., Gang, W., Huaiming, W.: Cooperation models for service oriented multi-agent system. In: Proceedings of the 2004 ACM symposium on Applied Computing, pp. 510–511. ACM Press, New York (2004)
27. Ramírez, E., Brena, R.: Multi-Agent Systems Integration in Enterprise Environments Using Web Services. In: Agent and Web Service Technologies in Virtual Enterprises, Information Science Reference, pp. 174–189. Hershey, New York (2008)
28. Huhns, M.N., Singh, M.P., Burstein, M., Decker, K., Durfee, E., Finin, T., Gasser, T.L., Goradia, H., Jennings, P.N., Lakkaraju, K., Nakashima, H., Parunak, H., Rosenschein, J.S., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T., Zavafa, L.: Research directions for service-oriented multiagent systems. IEEE Internet Computing 9(6), 65–70 (2005)
29. Jennings, N.R., Wooldridge, M.: Applications of intelligent agents. In: Agent technology: foundations, applications, and markets, pp. 3–28. Springer, New York (1998)
30. Mendes, J.M., Sousa, J., Leitão, P., Colombo, A.W., Restivo, F.: Event Router-Scheduler for the Modular Anatomy of Service-oriented Automation Components. In: Proceedings of the 6th CIRP International Conference on Intelligent Computation in Manufacturing Engineering (2008)
31. Mendes, J.M., Rodrigues, A., Leitão, P., Colombo, A.W., Restivo, F.: Distributed Control Patterns using Device Profile for Web Services. In: Proceedings of the 12th International IEEE EDOC Conference Workshop (2008)
32. Mendes, J.M., Bepperling, A., Pinto, J., Leitão, P., Restivo, F., Colombo, A.W.: Software Methodologies for the Engineering of Service-Oriented Industrial Automation: The Continuum Project. In: Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (to appear, 2009)

Usability of Multi-agent Based Control Systems in Industrial Automation

Ivanka Terzic[1], Alois Zoitl[1], Martijn Rooker[2], Thomas Strasser[2], Pavel Vrba[3], and Vladimir Mařík[3,4]

[1] Automation and Control Institute, Vienna University of Technology, Gusshausstr. 27-29 I 376, 1040 Vienna, Austria
{terzic,zoitl}@acin.tuwien.ac.at
[2] PROFACTOR GmbH, Im Stadtgut A2, 4407 Steyr-Gleink Austria
{Martijn.Rooker,Thomas.Strasser}@profactor.at
[3] Rockwell Automation Research Center, Pekarska 695/10a, 15500 Prague, Czech Republic
{pvrba,vmarik}@ra.rockwell.com
[4] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague 6, Czech Republic
marik@labe.felk.cvut.cz

Abstract. A future adaptive and reconfigurable manufacturing system has to address, beside the requirements for agile and fast reaction to sudden and unpredictable changes in production demands, also the problem of user acceptance and understanding of such highly intelligent reconfigurable systems. Thus, the usability is a major criterion for the design of future manufacturing systems. The design and development of both the architecture for simplifying of development of such systems and the advanced diagnostic, monitoring, and human-interaction system will provide an easy maintenance and increased acceptance of such systems.

Keywords: Usability, Multi-Agent Architecture, Automation, Diagnosis, User Acceptance.

1 Introduction

The mass production area was predominated by big investments for customized equipment, in order to produce faster and cheaper large quantities of identical or similar products. Even small changes in the product design or any failure in such a production system caused the production to stop and required a work- and time-intensive reprogramming of the system. The changing environment of today's market raises the need for new production planning and production control models and requires new approaches for production lines and intelligent machines to provide stability, sustainability, and economy under such production conditions. Being agile is important for reacting fast to sudden and unpredictable requirement changes with minimum risk. Adaptive, reconfigurable and modularly structured manufacturing systems address these requirements allowing machines and plants to flexibly adapt themselves to changing demands and interact with each other to fulfill the overall production goals.

While physical components of such systems are available, the implementation of reconfigurable systems in the manufacturing industry is hindered by the lack of knowledge-based methods and intelligent tools for their optimal deployment and control of their operation. Currently available methods and tools for the deployment and operation of such manufacturing systems are mostly based on traditional techniques applied in flexible manufacturing systems and are quite straightforward, addressing specific problems, lacking intelligence and learning capabilities. Moreover, their application takes place off-line, requiring significant down times as well as human interference. In order to reach the full potential of such systems, the adaptation of the system's performance to an optimal manufacturing solution for the appointed task needs to take place autonomously, in real-time and with as less human involvement as possible. Thus, the development of an autonomous intelligent governing system for adaptive modular reconfigurable manufacturing systems is of outmost importance.

But seen from today's standpoint of stakeholders in automation, the engineering of such manufacturing systems is too complex and their maintenance and service is presumed to be not manageable from outside anymore. Therefore, the usability should be a main criterion for the design of such highly reconfigurable intelligent manufacturing systems. In the future, it will not only be necessary to have adaptive reconfigurable manufacturing systems with the fast set up and implementation but also a production system that will be accepted by a user.

Such autonomous manufacturing systems with a modicum of human involvement have to be able, besides the performing of scheduled operation, to recognize possible problems and prevent them or, if not possible, to react on in an appropriate way and treat the failures. The on-line built-in diagnostic system is therefore crucial for improving the performance of such a system, but the static and pre-programmed diagnostics is pointless in that case [1]. It is impossible to preview all the critical situations which might occur in the life cycle of the system and in its possible configuration [2].

Thus, the design and development of both the architecture for simplifying of development of such systems and the advanced diagnostics, monitoring and human-interaction will provide easy maintenance and increase the acceptance of such systems by providing the transparent insights into the functionality of its control part. Within the frame of this paper a potential approach – currently in the early conceptual phase – for improved the industrial acceptance and use of modern control systems for reconfigurable manufacturing systems is given.

The paper is organized as follows: section 2 provides an overview of the state of the art in the agent-based control and describes the problem statement. A possible approach to overcome the shortcomings in the agent-based control is given in section 3. The summary and conclusions are provided in section 4.

2 State of the Art and Problem Statement in Multi-agent Control

The multi-agent control approach is widely used for developing and designing the adaptive, reconfigurable, and modular structured manufacturing systems, which allow machines and plants to flexibly adapt themselves to the changing demands and interact with each other to fulfill the overall production goals. It is already well known as a naturally suited technology for designing and implementing of dynamic and intelligent production systems [3]. To achieve a quick reconfiguration of manufacturing systems and to avoid a time- and work-intensive reprogramming, the use of the Plug & Play approach in multi-agent architectures is desirable [4]. For most of the existing multi-agent architectures a modular representation of production system components is the first effort in order to achieve this goal. Coalition Based Approach for Shop-floor Agility (COBASA) [5] and Actor-based Assembly Systems (ABAS) [6] can be named as examples for such a modular negotiation-based multi-agent architecture. A holonic approach, applied in ADACOR [7], PROSA [8] and MAST [9], are another approved architectures for intelligent manufacturing systems where a holon or holonic agent is a representation of a manufacturing component.

Tichý et al. present the Autonomous Cooperative System (ACS) architecture targeted to distributed industrial control applications [10]. The main aim is to integrate the low-level control ensuring the real-time responsiveness and the intelligent agents featuring more sophisticated algorithms for planning, communication and diagnostics. The common PLCs (Programmable Logical Controllers) provide the hardware infrastructure for running both the PLC programs (mainly ladder logic based) and the C++ agents in a parallel fashion. Important aspect to mention in the context of this paper is that each agent contains a Diagnostic Module, which task is to monitor the health of the attached hardware equipment. If the Diagnostic Module detects a failure of the equipment, the agent starts to negotiate with other agents about reconfiguring the control system such as the alternative solution utilizing redundant resources is provided.

In all of these modular multi-agent control systems the agents' information processing and reasoning are "hard-coded" in the agents' behaviors. The application of ontologies for knowledge representation, sharing and high-level reasoning can be seen as the next major step ahead towards flexibility in the agent-based control systems [11]. The ontologies are necessary to enable knowledge driven configuration and reconfiguration in an extensible manner. One of the first attempts to utilize ontologies in multi-agent systems for manufacturing is the work done by Obitko and Mařík [12]. It is pointed out that the usual approach, when manufacturing ontologies are expressed in XML/DTD and the semantics is described only informally in a natural language, is not sufficient. It is necessary to have the formal description of the domain of interest that would be understandable to computer agents. These authors argue for utilizing Web Ontology Language (OWL) for expressing semantics and provide an ontology example from the transportation domain together with the discussion on the possibilities of translations between different ontologies [13].

Other ontologies describing specific areas of manufacturing domain are for instance the OZONE ontology [14], the Enterprise Ontology [15], or the "Machine Shop Information Model" [16]. The MASON ontology could be seen as a draft of an overall ontology for the manufacturing domain [17].

Ontologies and knowledge bases utilizing shared ontologies can be used for (re-) configuring production systems instead of reprogramming agents or even the whole control systems in order to adapt a production to the new conditions. Such an ontology-based multi-agent system is implemented e.g. at the NovaFlex shop floor assembly domain of Intelligent Robotic Centre in UNINOVA, Lisbon [18]. In the frame of the EUREKA Factory E-RACE project (E-RACE, 2004), Web-space for decision-making support and design of reconfigurable assembly systems was developed based on knowledge domains. The examples of domains specified with ontologies are product, process, concept, and embodiment [19]. An ontology-driven multi-agent control architecture within the transportation domain based on simulation of an assembly process is presented in [20]. Also Evolvable Assembly System (EAS) referred in [21] describes an ontology-based multi-agent control concept for assembly based on finely granulated modules representing manufacturing components. The vision of EAS is that the system will be aware of its current capabilities and capacities by receiving the order from the user, and that it will be able to propose the modification or extension by plugging equipment modules available from different vendors [2]. The key aspects of deployment of semantics and ontologies in distributed manufacturing control system are summarized in [22].

Common for all above named architectures and approaches is that the problem of user acceptance and understanding of “intelligent” control systems in automation industry was not explicitly focused. But it seems to be one of the key issues developers of reconfigurable multi-agent based systems are faced with [23].

Rockwell Automation, Inc. was confronted with this problem during its agentification project aimed at the steel rod bar mill for BHP Billiton, Melbourne. In order to increase the machine utilization, the agent-based control system was developed for dynamic selection and configuration of production resources. However, due to the anxiety about possible wrong decisions of the multi-agent control system that could lead to damage of equipment or production loss, BHP Billiton decided not to use it for direct physical control. Instead, the agent control system provided on-line recommendations for resource configurations and control actions, while the final decision about their execution was left to the plant operator. Regardless of this policy, the agent system proved to work well in all test cases. BHP Billiton was also reluctant to use a new intelligent control system in production because of the lack of skilled personnel able to maintain such a system [24]. DaimlerChrysler's multi-agent project Production 2000+ faced the similar issues. This multi-agent system targeted to control a flexible and robust production system for large-volume power-train manufacturing, and has been tested by simulation utilizing real production data (e.g., processing times, real disturbance characteristics of existing production systems). In order to validate simulation test results, DaimlerChrysler installed this production system as a bypass to an existing large-volume manufacturing line for cylinder heads in its plant in Stuttgart-Untertürkheim (Germany) [25]. However, the test phase was aborted after about one year, because operators were overstrained with the new flexible production system and especially with the problem that they were not able to determine if the plant is working correctly or not.

In principle, there are many obstacles to widespread adoption of agent-based technology in industry [26], namely

- *thinking of engineers* who were educated to consider centralized approaches only;
- *risks* as the industry environment is afraid of emergent behavior of MAS without any central unit – there are no formal algorithms or procedures guaranteeing that the MAS systems would behave as desired;
- *costs* – the immediate costs of introducing of the agent-based technology are higher than in the “classical” systems;

- *vendor-centric view* – until the end-users are able to develop and maintain MAS by themselves in a straightforward way, these systems are accepted with very serious difficulties, the end-user wants to keep things under control and to understand the solutions fully.

3 Proposed Approach for an Improved Multi-agent Control System

In order to increase the industrial acceptance and use of multi-agent systems for adaptive reconfigurable production systems, we propose a twofold approach consisting of:

- a modular, knowledge-based multi-agent architecture for simplifying the development of multi-agent systems, and
- a diagnostic and user interaction infrastructure for improving the plant operator interaction.

In order to simplify the development of Multi-Agent System (MAS), in our ontology driven multi-agent architecture, the system modularity will achieve very fine granularity on all the considered levels – that means on functional, physical, and software implementation levels. In order to leverage the advantages of using a building block principle in modularly structured production systems, small physical components together with local intelligence are capable to build any system configuration and to modify it. Besides the plug-ability of physical manufacturing agent modules, the usability and rapid (re-)configuration of the system is also assured through the plug-ability of finely granular behavior modules at the reasoning level. The proposed knowledge-based MAS provides a very flexible cognitive architecture, where new behavior or new features of an agent can be assembled from such explicitly defined behavior modules using a building block principle. That means the fine module granularity is given at the level of physical components as well as at the reasoning level of agents. Therefore, besides the plug-ability of manufacturing agent modules, our framework provides quite a flexible cognitive architecture with a mechanism of explicitly defined behavior modules, enabling the plug-ability at all the architecture's level. The new behavior or new features of an agent can be assembled from such explicitly defined behavior modules.

In order to address the issues of low trust in multi-agent industrial control architectures and anxiety from unpredictable and unmanageable agent behavior, we propose an architecture, where the human operator acts as a part – an agent – of the multi-agent control system. A diagnostic system identifying the current system state and notifying on wrong system behavior is crucial for improving the confidence in multi-agent control systems [1, 2]. Human Machine Interface embodies agents that collect real-time data, diagnostic events and decisions of particular agents and make this information accessible to the human actor agents. Depending on the environment's requirements or diagnostic results, the meta-agents give a recommendation or direction to the operator and, vice versa, accept recommendations or decisions from the operator. The attention should be paid to possible dangerous states requiring operator's attention as well as to correction of possible dangerous or wrong decisions of the operator.

3.1 Modular Knowledge-Based Agent Architecture

A main issue of the existing multi-agent approaches is that although the overall multi-agent system is very modular and flexible, the specific agents themselves are monolithic and specially tailored for their purpose at hand. Also high skills of programming are required from developers in order to develop or to modify agents. Therefore we propose a generic modular agent architecture which can be configured for the given specific functionality of the agent (see Fig. 1).

The proposed agent architecture is based on a hybrid two-tier architecture, where the functional layer implements the basic functionality, and the reasoning layer serves for deliberation. On the functional layer, a limited set of behaviors defines how the functional layer reacts within the environment. It basically controls the sensors and actuators. Any change of sensor data may cause a reaction in this layer. This layer can be given low-level tasks by the reasoning layer. A default behavior, relying upon no perceptual information, is sanctioned if no other action is initiated. This causes the agent to continue accomplishing its current task. On this layer, it is also possible to accomplish certain simple goal-oriented steps such as to react to internal and external states.

The knowledge-base is a fundamental part of the agent as it captures the actual state of the surrounding environment as well as the agent itself in a semantic context. It means that the data the agent obtains by sensors or by communication with other agents are given semantic meaning by associating the facts with matching elements in the knowledge structure provided by the ontology. The important aspect is the reflectivity – the knowledge base representing the agent's data model of the reality contains the notion of itself, thus enabling the agent to reason about itself in the context of relations to other entities in the environment. Self-reflectivity achieved in this way, represents an important step towards higher autonomy of agents [27].

Let us have a simple example from the transportation domain. The ontology defines for instance classes representing a *conveyor*, *diverter* and relation *connectedTo*. In a particular scenario, the knowledge base of the conveyor belt agent *cb14*, which connects the diverter *d13* with the diverter *d15* contains an instance *cb14* of the conveyor class representing the agent itself and two instances *d13* and *d15* of the diverter class linked with the directed *connectedTo* relation to the instance *cb14*. This knowledge can be used by conveyor agent for reasoning about itself. For instance, when the diverter *d15* connected to the end of the conveyor sends a message about its failure, the conveyor agent updates its knowlege base with this new fact (by attaching failure label to object d15 in knowlege base). Then, following the *connectedTo* relation from *d15* to *cb14* (itself) the agent realizes that he is also affected by the failure and that an action is needed (e.g., to stop itself so that items currently on the conveyor are not fed to the failed diverter).

The novelty in this new agent architecture is how the knowledge-base is maintained and how it is integrated with the activities performed by the agent. The Behavior Management Component manages so called behavior modules that associate particular declarative knowledge with procedural knowledge representing the behavioral aspects of the agent. When a new fact is stored in the knowledge base the agent executes associated rule-based behavior as an action required to be taken as a response to new information. The proposed architecture allows to modify the behavior

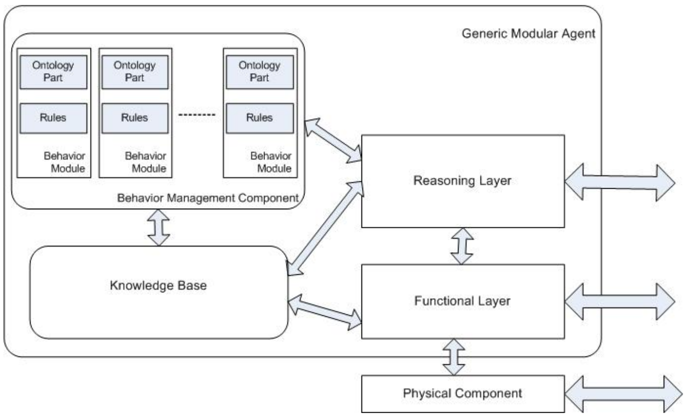


Fig. 1. Overview of the generic modular agent

modules of the agent on-line; the agent can receive new modules in order to extend or modify its knowledge, both declarative and procedural.

In general, many of the behavior modules will be standard components and only a small part has to be newly developed. Therefore, a basic set of behavior modules handling standard tasks of agents should be developed. Such standard tasks will be for example interaction with other agents, interaction with low level control, planning, scheduling, self-diagnostics, etc.

The proposed architecture of the generic agent can be seen as an extension of the acquaintance model architecture [28], which was widely used in ProPlanT and Ex-PlanTech family of production planning and scheduling systems [29], where the three knowledge bases contained the generic rules, the list of used rule instances and the database of plans/schedules, respectively.

The advantage of the approach presented in this paper is that the behaviors (consisting of generic rules, but also knowledge ontologies in addition) are represented by modular, functionally self-contained entities – building blocks – allowing simple assembling of specific behavior in a similar manner as the Lego^{TM} blocks. Both the high degree of modularity and the application of ontologies do differentiate our solution from the acquaintance model approach. Due to this fact it is easier for the user to understand, to program, and, if needed, to manipulate such a single component than a whole complex algorithm of a typically programmed agent.

3.2 Diagnostic and User Interaction Infrastructure

In order to gain confidence in the new multi-agent technology among the managers and stakeholders in industry, it is necessary to design and develop an advanced diagnostic, monitoring and human-interaction system that would provide transparent insights into the functionality of the agent control system together with means for easy maintenance of such a system. An important factor to be considered is that the personnel that use these systems might not have any knowledge about the internal functionality of agents, their behaviors and decision-making, inter-agent negotiations, etc. Thus these "behind-the-scene" processes should stay hidden for the user; however, they must be translated into a format comprehensible to the user. This is an especially challenging task for agent-based systems where the reconfiguration of production processes occurs at first sight randomly and unpredictably. This is caused by the autonomy- and interaction-based character of agent systems, where any event (e.g., failure) diagnosed by one agent starts a cascade of messages sent among agents in order to take a corresponding action, either in a collaborative or competitive spirit.

We propose a general framework for providing clear explanations of the diagnosed events and consequent actions of the agent system. The framework is based on a hierarchy of meta-agents that monitor the activity of agents at particular level of the manufacturing enterprise and provide this information in a meaningful form to the user. At each hierarchical level, an appropriate knowledge bundle with relevant information granularity is provided to the user - from the highest ERP levels with overall business performance insights down to the real-time control level with the information for plant operators about which particular sensor notified the failure (see Fig. 2).

Beside the detection and correction of the failure, which has already occurred, the future production systems should be able to predict possible failure in order to act in a preventive way. That means the production system should be able not only to react to the failures such as collision, mechanical malfunction etc., but also meet the problem of wear-out and preventive maintenance. Such service requires permanent monitoring, analysis, and evaluation of the status of manufacturing resources and the whole production system as well as periodic reports to the users on different hierarchical levels. Due to this, service and predictive maintenance of the manufacturing resource or system can be accomplished not only in fixed time intervals, but earlier or later depending on the detected status of the manufacturing resource in a shop floor. That means the failure detection and correction should be performed both locally by the agents representing manufacturing resources, based on their local knowledge and

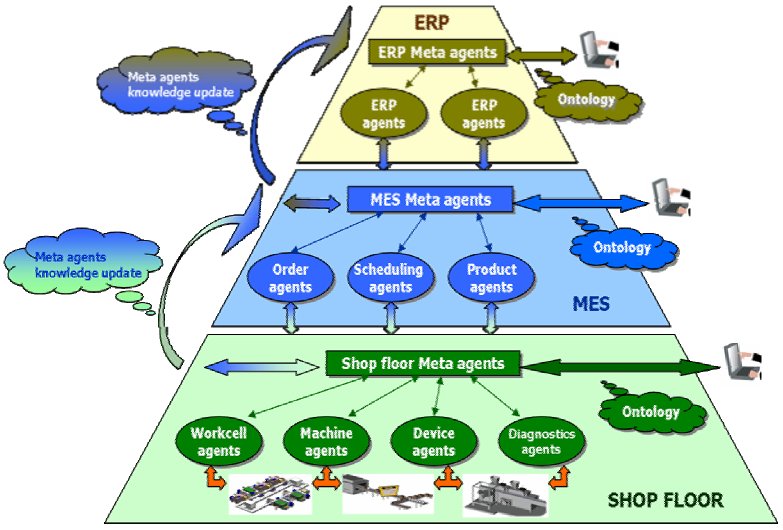


Fig. 2. Architecture of the diagnostic and user interaction infrastructure

information, as well as by agents representing production functions (such as the order or the supply) in a higher hierarchical level of the manufacturing enterprise, monitoring the possible failures emerged due to interaction of manufacturing resources. This is ensured by the cross-layer interaction between meta-gents as shown in Fig. 2.

Therefore each agent representing a certain manufacturing resource should be able to make diagnostic conclusions about its status, actions, behaviors etc. and to make its own decision if it is able to accomplish any correction of the diagnosed failure or not. That means the agent has to monitor itself and to reason about its state. In our generic modular agent architecture (see Fig. 1) this is the task of the reasoning layer. The agent knows its exact behavior in order to achieve its goal and to fulfill a given task. It compares the expected behavior with the observed behavior. If any discrepancy between the observed and expected behavior appears, the reasoning layer is expected to try to explain this discrepancy and to detect the possible failure. If possible, the reasoning layer starts to correct the failure; otherwise human intervention will be requested through the meta-gents at given layer. At the same time, the production system adapts itself to the new conditions by interacting and negotiating among its agents in order to maintain the production. If, for example, the malfunction of a conveyer is detected, the reasoning layer of the conveyer agent tries to explain and detect the failure (e.g., the malfunction of the conveyer motor). Contemporarily, the transport system adapts itself to the new conditions: the relevant pallets will be rerouted; an error report will be generated and sent to the Shop Floor meta-agent and the diagnostic agent. Depending on the severity of disturbance, the meta-agent decides to inform the operator about the disturbance immediately – asking for the operator's acknowledgement to start with error correction measures – or to postpone the provision of the information to a regular error summary report. If the conveyer agent comes to the conclusion that it is not able to repair the disturbance, it will contact its meta-agent and require human intervention. Also if the malfunction detected by an agent is unknown to him, the agent contacts the diagnostic agent at first, containing history of the disturbances and malfunctions in the system for some period of time, asking for help. The direct human user intervention will be requested only if the malfunction is also unknown to the diagnostic agent and the production system is incapable to solve the problem by itself. The same diagnostic principle [30] used by the agent representing a manufacturing resource can be applied to agents representing production functions on a higher hierarchical level of the manufacturing enterprise.

The goal is to have a diagnostic system that can achieve some attributes and abilities as named in [31]: early fault detection and diagnosis, ability to differentiate among different failures, robustness, ability to identify multiple faults, ability to provide explanations on how the fault originated and propagated to the current situation, adaptability, ability to decide, whether the process is normal or abnormal and if abnormal, whether the cause is a known malfunction or an unknown, novel malfunction, etc.

The fact that in our ontology-driven MAS the manufacturing system is represented by a combination of agents representing all production participants, allows the human user to interact with a system at appropriate level of manufacturing enterprise, to get or provide the information needed or to intervene if required. Information and knowledge exchange among meta-agents and users is based on application of a modular ontology providing unambiguous semantic interpretation of states, events and actions occurring in the production system. Based on the login data of the user the meta-agent of particular level of manufacturing enterprise will be contacted. The interaction, information and knowledge exchange between the user and the meta-agent do depend on user's access authority. So, any deeper understanding of the complexity and functionality on all levels of the underlying system is not required. The meta-agent provides relevant knowledge and information to the user.

The Human Machine Interface design in terms of Graphical User Interface in order to increase the user-friendliness of high intelligent manufacturing system is another important issue. As a paradigm, the experience and principles of Usage-Centered Design [32] or User-Centered Design [33] can be used and applied.

4 Summary and Conclusions

The current industrial agent-based solutions suffer from their quite low acceptance on different management levels of companies: the low acceptance can be seen by the company management, engineering community, and operators. There are many reasons for it, namely lack of "distributed thinking", increased investment costs and risks connected with such solutions [26]. In order to increase the industrial acceptance and use of multi-agent systems for adaptive reconfigurable production systems, we have proposed a multi-agent solution based on a modular, knowledge-based architecture of individual agents. This architecture stresses the autonomy of the agents and efficient and friendly communication with the users. Highly modular internal agent's structure exploring knowledge ontologies provides basic diagnostic features inherently and contains a user interaction subsystem for simplifying the development and enhancing the plant operator interaction.

The modularity of the proposed architecture is aimed at modular composition of models of agent´s behavior, capabilities of self-reflection [27] and enhancing the agent´s performance by making use of semantics and knowledge ontologies.

Through our modular knowledge-based agent architecture, the control engineer can more efficiently assemble the needed agent's behavior. Furthermore, the proposed architecture provides an efficient support for reconfiguring agent behaviors. It is able to foresee also several levels of diagnostic conclusions and different kinds of knowledge and facilitates the user interaction. Integrating our agent-based diagnostic approach and appropriate Human Machine Interface, the plant operator is expected to be able to interact better with the control system and more thoroughly understand the system's state. This will increase the operator oversight of the plant and greatly reduce wrong operator interactions.

References

1. Dencker, K., Stahre, J., Grondahl, P., Martensson, L., Lundholm, T., Johansson, C.: An Approach to Proactive Assembly Systems: Towards Competitive Assembly Systems. In: Proc. of the 2007 IEEE Int. Symp. on Assembly and Manufacturing Ann Arbor, Michigan, USA (2007)
2. Frei, R., Ribeiro, L., Barata, J., Semere, D.: Evolvable Assembly Systems: Towards User Friendly Manufacturing. In: IEEE Int. Symp. on Assembly and Manufacturing Ann Arbor, Michigan, USA (2007)

3. Bussman, S., Jennings, N.R., Wooldridge, M.: Multiagent System for Manufacturing Control. A Design Methodology. Springer Series on Agent Technology (2004)
4. Frei, R., Serugendo, G.D.M., Barata, J.: Designing Self-Organization for Evolvable Assembly Systems. In: Int. Conf. on Self-Adaptive and Self-Organizing Systems, pp. 97–106 (2008)
5. Barata, J.: The Cobasa Architecture as an Answer to Shop Floor Agility. Manufacturing the Future. Concepts - Technologies - Visions, p. 908, ARS/plV, Germany (2006)
6. Lastra, M.J.L., Torres, E.L., Colombo, A.W.: A 3D Visualization and Simulation Framework for Intelligent Physical Agents. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS, vol. 3593, pp. 23–38. Springer, Heidelberg (2005)
7. Leitao, P., Restivo, F.: ADACOR: A Holonic Architecture For Agile and Adaptive Manufacturing Control. Computers in Industry 57(2), 121–130 (2006)
8. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeter, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers in Industry 37(3), 255–274 (1998)
9. Vrba, P., Mařík, V., Merdan, M.: Physical Deployment of Agent-based Industrial Control Solutions: MAST Story. In: Talk: IEEE SMC Int. Conf. on Distributed Human-Machine Systems (DHMS), Athens, Greece, pp. 133–139 (2008)
10. Tichý, P., Šlechta, P., Maturana, F., Balasubramanian, S.: Industrial MAS for Planning and Control. In: Mařík, V., Štěpánková, O., Krautwurmová, H., Luck, M. (eds.) ACAI 2001, EASSS 2001, AEMAS 2001, and HoloMAS 2001. LNCS (LNAI), vol. 2322, pp. 280–295. Springer, Heidelberg (2002)
11. Obitko, M., Mařík, V.: Ontologies for Multi-Agent Systems in Manufacturing Domain. In: Proc. of the 13th Int. Workshop on Database and Expert Systems Applications DEXA 2002 (2002)
12. Obitko, M., Mařík, V.: Adding OWL Semantics to Ontologies Used in Multi-Agent Systems for Manufacturing. In: Mařík, V., McFarlane, D.C., Valckenaers, P. (eds.) HoloMAS 2003. LNCS (LNAI), vol. 2744, pp. 189–200. Springer, Heidelberg (2003)
13. Obitko, M., Mařík, V.: Integrating Transportation Ontologies Using Semantic Web Languages. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS (LNAI), vol. 3593, pp. 99–110. Springer, Heidelberg (2005)
14. Smith, S., Becker, M.: An Ontology for Constructing Scheduling Systems. In: Working Notes of 1997 AAAI Symp. on Ontological Engineering, AAAI Press, Menlo Park (1997)
15. Uschold, M., King, M., Moralee, S., Zorgios, Y.: The Enterprise Ontology The Knowledge Engineering Review. In: Uschold, M., Tate, A. (eds.) Special Issue on Putting Ontologies to Use, vol. 13 (1998)
16. McLean, C., Lee, Y.T., Shao, G., Riddick, F.: Shop Data Model and Interface Specification, NISTIR 7198. National Institute of Standards and Technology, Gaithersburg (2005)
17. Lemaignan, S., Siadat, A., Dantan, J.-Y., Semenenko, A.: MASON: A Proposal For An Ontology Of Manufacturing Domain. In: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications DIS 2006, pp. 195–200. IEEE Computer Society Press, Los Alamitos (2006)
18. Candido, G., Barata, J.: A Multiagent Control System for Shop Floor Assembly. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 293–302. Springer, Heidelberg (2007)
19. Lohse, L., Ratchev, S., Valtchanov, G.: Towards Web-Enabled Design of Modular Assembly Systems. Assembly Automation 24(3), 270–279 (2004)
20. Merdan, M., Koppensteiner, G., Hegny, I., Favre-Bulle, B.: Application of an Ontology in a Transport Domain. In: Proc. IEEE Int. Conf. on Industrial Technology (2008)

21. Frei, R., Barata, J., Onori, M.: Evolvable Production Systems - Context and Implications. In: ISIE 2007 - IEEE Int. Symp. on Industrial Electronics, Vigo, Spain (2007)
22. Obitko, M., Vrba, P., Mařík, V., Radakovič, M.: Semantics in Industrial Distributed Systems. In: Proc. of the 17th IFAC World Congress, Seoul, Korea, pp. 13880–13887 (2008)
23. Terzic, I., Zoitl, A., Favre, B., Strasser, T.: A survey of Distributed Intelligence in Automation in European Industry, Research and Market. In: Proc. IEEE Int. Conf. on Emerging Technologies and Factory Automation, pp. 221–228 (2008)
24. Mařík, V., Vrba, P., Hall, K.H., Maturana, F.P.: Rockwell Automation Agents for Manufacturing. In: Proceedings of the Industry Track of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, Utrecht, Netherlands, pp. 107–113 (2005)
25. Bussmann, S., Schild, K.: An Agent-Based Approach to the Control of Flexible Production Systems. In: Proc. of the 8th IEEE Int. Conf. on Emergent Technologies and Factory Automation, France, vol. 2, pp. 481–488 (2001)
26. Pěchouček, M., Mařík, V.: Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies. J. of Autonom. Agents and Multi-agents Systems, 397–431 (2008)
27. Rehák, M., Tožička, J., Pěchouček, M., Železný, F., Rollo, M.: An Abstract Architecture for Computational Reflection in Multi-Agent Systems. In: Intelligent Agent Technology, pp. 128–131. IEEE Computer Society Press, Los Alamitos (2005)
28. Pěchouček, M., Mařík, V., Štěpánková, O.: Role of Acquaintance Models in Agent-based Production Planning Systems. In: Klusch, M., Kerschberg, L. (eds.) CIA 2000. LNCS, vol. 1860, pp. 179–190. Springer, Heidelberg (2000)
29. Pěchouček, M., Rehák, M., Charvát, P., Vlček, T., Kolář, M.: Agent Based Approach to Mass Oriented Production Planning. IEEE Transactions on Systems, Man, and Cybernetics - Part C 37(3), 553–560 (2007)
30. Clancy, J.D.: Qualitative, Model-Based Diagnosis of Complex Physical Devices. In: 1998 IEEE Int. Conf. on Systems, Man, and Cybernetics, San Diego, CA, USA, vol. 3, pp. 3012–3019 (1998)
31. Dash, S., Venkatasubramanian, V.: Challenges in the Industrial Applications of Fault Diagnostic Systems. Computers and Chemical Engineering 24, 785–791 (2000)
32. Constantine, L.L., Biddle, R., Noble, J.: Usage-Centered Design and Software Engineering: Models for Integration. In: Int. Conf. on Software Engineering, Portland (2003)
33. Keinonen, T.: User-Centered Design and Fundamental Need. In: Proc. of the 5th Nordic Conference on Human-Computer interaction: Building Bridges. NordiCHI 2008, vol. 358, pp. 211–219. ACM, New York (2008)

An Organizational Knowledge Ontology for Automotive Supply Chains

Bernd Hellingrath[1], Markus Witthaut[2], Carsten Böhle[3,⋆], and Stephan Brügger[3]

[1] University of Münster, Germany
hellingrath@ercis.uni-muenster.de
[2] Fraunhofer Institute for Material Flow and Logistics, Dortmund, Germany
markus.witthaut@iml.fraunhofer.de
[3] Heinz Nixdorf Institute, University of Paderborn, Germany
carsten.boehle@hni.uni-paderborn.de

Abstract. The currently completed ILIPT (Intelligent Logistics for Innovative Product Technologies) project was concerned with the concept of the "5 day car" (a customized car that is delivered within five days after its ordering) and encompassed extensive research on the required production and logistics network structures and processes. As car manufacturers in the automotive industry (commonly referred to as OEMs) rely heavily on their suppliers, the major challenge lies in the organization of inter-enterprise cooperation supported by information systems (IS) in an efficient manner. A common understanding of supply chain concepts is indispensable for this. Ontologies as formal representations of concepts can be used as a semantic basis for cooperation. Relevant results from ILIPT are presented followed by a concept as well as a prototype of how to transfer the theoretical findings to a practical implementation, in this case a multi-agent system.

Keywords: Ontology, Supply Chain Management, Multi-Agent System.

1 Introduction

Research concerning ontologies is in most cases embedded in the context of the semantic web. There are, however, several fields of application in the manufacturing industry. The automotive industry is a good example as it consists of a huge network of suppliers and OEMs interlinked by material and information flows. The control of the flow of goods, information, and funds within this network is investigated by the field of Supply Chain Management. The basics of ontologies and Supply Chain Management will be given here along with the motivation of why to bring these two topics together.

The rest of the paper is organized as follows. Section 2 summarizes previous approaches. Section 3 describes the relevant topics from ILIPT in detail. Section 4 lists the requirements that stem from automotive supply networks and how

⋆ Corresponding author.

ILIPT results can be transferred for a solution. Section 5 gives the results, i.e. the different models that were developed. Section 6 decribes the application in a multi-agent system. Section 7 concludes the paper with an outlook on further research.

1.1 Ontologies

An ontology in general is defined as the "formal, explicit specification of a shared conceptualization" [1]. It is constructed for certain domains and captures the knowledge held therein. Ontologies can be designed as solutions to specific challenges. In the supply network domain, a distinction into organizational ontologies and problem ontologies[1] is reasonable. The former describes organizational knowledge, defined as "the capability members of an organization have developed to draw distinctions in the process of carrying out their work, in particular concrete contexts, by enacting sets of generalizations whose application depends on historically evolved collective understandings" [4], or simply, the knowledge that "helps to understand the domain requirements" [5]. This knowledge contains structures like products and processes and enables organizations to make use of their resources. As opposed to that, problem solving knowledge formalizes the meaning of problems and sets it into relation to the concepts provided in the organizational ontology. A problem solving ontology is the basis for communication and collaborative problem solving. This paper will focus on the organizational aspect as the problem ontology has to be built on top of it and will be subject to further research.

1.2 Supply Chain Management

Supply Chain Management (SCM) denotes a field of research which is concerned with the development of tools and methods for the collaboration within the supply network. More precisely: SCM is the integrated, process oriented design, planning, and control of the flow of material, information, and cash along the whole value chain, ranging from the consumer to the raw material producer, aiming at enhanced customer orientation, synchronization of production and demand, flexible and demand oriented production, and the reduction of inventories along the supply chain [6].

1.3 Motivation

Each enterprise within a supply network holds its own concepts and data models. This is fine as long as collaboration consists of placing orders and receiving goods only. As soon as collaboration becomes closer, the lack of a common understanding hinders efficient inter-enterprise processes. Ontologies can help at that point since they force every participant to formalize their knowledge. Once

[1] The terms "domain ontology" and "application ontology" or "task ontology" can also be found in literature (cf. [2] [3]).

this is done, IS can be built that make use of them. In contrast to that, there is criticism that "literature falls short in defining supply chain knowledge and information, their characteristics and their interdependence" [7]. The reasons why we chose ontologies instead of relational database schemes is that we believe the rich semantics will foster cooperative development and that most concepts can be modeled easier and more straightforward. The aim of this paper is to present an ontology that describes the field of automotive supply chains in sufficient detail so that an IS is able to make use of it. The future goal is to support the design of cooperating IS that will facilitate business processes between companies in order to provide improved production and logistics plans avoiding capacity shortages.

2 Ontologies in Supply Chain Management

Little research exists on this topic. Chandra [5] [8] [9] [10] has frequently published in this field. One of his papers [8] presents a framework for the development of a taxonomy along with an example for a production system. It is constructed by concretizing a general structure and enriching it with more specific information. No hints are given on if and how reoccurring structures such as the bill of material (BOM) are represented. In the follow-up paper [5], the distinction between problem knowledge and organizational knowledge is introduced. The resulting ontology is presented as an extension of the taxonomy. The latest work points out the integration with IS, especially multi-agent systems, where agents can be used to interact with other agents based on their knowledge to plan and execute a supply chain. A proof of concept has yet to be delivered. Ye et al. [11] claim that interoperability on the semantic level is the key for successful business process integration. They construct an ontology that consists of different parties that play a certain role within a supply chain. Parties can perform activities which require resources. Moreover, each party has objectives and realizes an individual performance. Processes are not explicitly contained but can be described as complex activities. Blomqvist et al. [12] developed an ontology aimed at small and medium sized enterprises. Their approach is deductive, i.e. they split up a supply chain into its constituent parts. The paper focuses on the task of supply chain configuration. Problem solving ontologies are mentioned but not further examined. The authors bring up the problem that domain specific ontologies are not only implementations of a general ontology, but that they often contain data in much more detail so that mapping is not sufficient and a refinement has to be made. Mollaghasemi et al. [7] [13] state that until 2004 "there has not been a lot of use of ontologies in supply chain management research or in enterprise engineering research in a broader sense". They describe an ontology based on the Supply Chain Operations Reference Model (SCOR). The result is a process centric ontology that does not capture facilities, BOMs, or other entities and their attributes. Umeda and Lee [14] analyze what processes and data are necessary in order to conduct cooperative planning processes in a supply chain, thereby taking the first step towards a problem ontology. Fox et al. [15] [16] have developed the TOVE ontology which is limited to a single enterprise and has been discontinued since 2001. All authors acknowledge the fact that members of a supply chain must agree on a common vocabulary in order to work together efficiently. The way to formalize this vocabulary has so far been a top-down approach where first a generic supply chain ontology is devised which in turn serves as a basis for domain specific specialization. This paper will present a bottom-up approach where a supply chain ontology is developed based on specific domain knowledge. A potential loss of generality is accepted in exchange for a hands-on ontology.

3 ILIPT

ILIPT (Intelligent Logistics for Innovative Product Technologies) [17] is a project sponsored by the European Union. Its objective is to find methods, structures, and technologies that make it possible to produce a customized car within a few days after being ordered. Besides research institutions also OEMs, suppliers, and logistics service providers are part of the consortium.

The ILIPT project is working on the requirements to move the European automotive industry from the "stock push" and "mass production" thinking of the last century to a stockless "build-to-order" (BTO) production strategy. It therefore aims at an order-specific final assembly and component production in the network as far as this is feasible regarding drastically reduced delivery target dates and cost effectiveness in respect to the aspired massive reduction of inventory in expensive cars and components. ILIPT investigates all relevant influencing factors enabling built-to-order production and logistics processes in the network. For this reason, ILIPT is working in the three major thematic areas of product structures, planning and execution processes in the production network together with the supporting IT systems and the methods for modeling and evaluating supply and distribution networks.

The target of a short order-to-delivery time under the requirement of a customer specific production requires highly capable processes for controlling the supply and distribution network. In the design of these processes, ILIPT follows the idea of planning and executing the network operations on the basis of final customer orders. The planning processes of ILIPT focus onto the improvement of the flexibility and the ability to respond to the supply and distribution network. Guiding principle is the decentralization of the planning processes on a collaboration basis, leaving the planning autonomy with the respective company, an indispensable requirement in the highly interwoven supply structures in the automotive industry. Network partners are only loosely coupled through the exchange of demand and capacity information. Decisions are made through iterative negotiation-based co-ordination of the different network partners within the boundaries of strategic general agreements. The execution processes of ILIPT include the order management, the order sequencing, and the control of the material flow. The latter processes take care for the monitoring of the material supply through build-to-stock suppliers and the handling of breakdowns and exceptions. Within the order management, the feasibility of delivering the requested car in a few days is already checked at the moment the order is entered in a configurator or ordering system. To allow this, the capacity situation of the whole network is continuously monitored. The realization of the described planning and execution processing is only possible with the development of supporting IT systems. To enable the communication among the network partners for the planning and execution processes, a so called Virtual-Order-Bank (VOB) has been developed in the ILIPT project. The VOB is an integrated order management and scheduling system which connects customers, dealers, OEM, suppliers and logistics service providers across the whole network. Basic idea of the VOB is the direct connection of customer demands and the capacity offers in production and logistics of the network companies, thus allowing the real-time synchronization of the network. Therefore, all incoming customer orders are balanced with the available capacities of the final assembly of the OEM, the suppliers, and logistics service providers in the built-to-order part of the network. The VOB manages and updates the capacity agreements and restrictions of all relevant companies. The amount of allocated capacities by accepted orders in relation to the agreed capacities is thus transparent for every built-to-order producer. On this basis, it can be checked directly with the entry of an order in how far it is satisfiable and the orders are booked to the necessary capacities at each plant and assembly period. If the requested delivery date is not feasible, the VOB determines an alternative date, communicating it to the customer, or informs the customer about the selected vehicle options delaying the delivery. In periods of under-utilization, the VOB determines in how far long-term or fleet orders can be postponed or brought forward to achieve the desired level of utilization in the considered time period. The special characteristic of the planning support given by the VOB is the fast and straightforward connection of the local planning systems used in the network in order to balance demand and capacity across several tiers in the network. The development of interoperability concepts and standards for the seamless information flow between the VOB and among the heterogeneous information systems is also a field of activity in the ILIPT project and is described in the following.

The requirements implied by the methodologies of new collaborative interaction in supply networks are not fulfilled by current information systems and the underlying infrastructure of the automotive industry. Heterogeneity of IS employed by enterprises arises on a large scale - computer environments, hardware and software platforms, programming languages, middleware systems, communication mechanisms and protocols, data formats, etc. For instance, supply chain partners currently depend on software applications exchanging information mostly on basis of proprietary data schemes and interfaces using non-standard transportation and application protocols. This hampers cross-company system integration in the context of ILIPT. Consequently, a common understanding of information exchanged between the partners of an automotive network is a mandatory constituent of collaborative planning and execution processes. Current practice for reaching this understanding is the application of business standards such as Odette or OAGIS (Open Applications Group Integration Specification) [18] which standardize messages - e.g. delivery forecasts or purchase orders - that are exchanged between the network partners. This approach has a major drawback: the semantics of the messages are described in an informal manner and leave room for interpretation. The consequence is that the meaning of a message or a message element is only informally described and not attached to the message itself. As a result, the establishment of an automated information exchange between a customer and supplier is both time consuming and error prone.

4 Design Methodology and Requirements

As was mentioned earlier, this work follows a new and unique approach which is different to previous efforts which tried to devise generic models and customize them for a certain application area in the field of supply chain management. Here, the purpose is known beforehand and the domain has been studied in detail in the ILIPT research project. From that, concepts and relations were already documented and had to be codified. Protégé [19] was used to model the ontology in OWL. Among the requirements for the formal model was the ability to describe different planning entities with different calendars, to have different levels of product descriptions, to make a distinction between built-to-stock and built-to-order suppliers, to reflect capacity information, and to integrate capacity adjustment measures. Although the application domain has been confined to automotive supply chains, scenarios can still differ widely. Certain building blocks might not be required in some scenarios whereas specialized data structures might miss in others, i.e. modularity is very important. Altogether the field of application is well known and the ontologies represent the insights drawn from the ILIPT project which supposedly make them useable in any kind of automotive supply chain project. Last, one important design step is to integrate extensions made in projects to the standard model in order to profit from new insights and to prevent forking.

5 Results

Based on the findings from ILIPT, five different models have been designed from which a planning entity can be built: sourcing model, resource model, adjustment measure model, demand constraint model, and time model. According to the bottom-up approach, these models are the least common denominator for the examined network. Specific implementations might need more structures and relations which can be added to this framework. The function of the models will be explained in the following. The ontology is available upon request from the authors. Only a brief summary of the models can be provided here.

5.1 Sourcing Model

The Sourcing Model (cf. figure 1) gives information about the products sold to customers and the parts procured from suppliers as well as the bilateral contracts

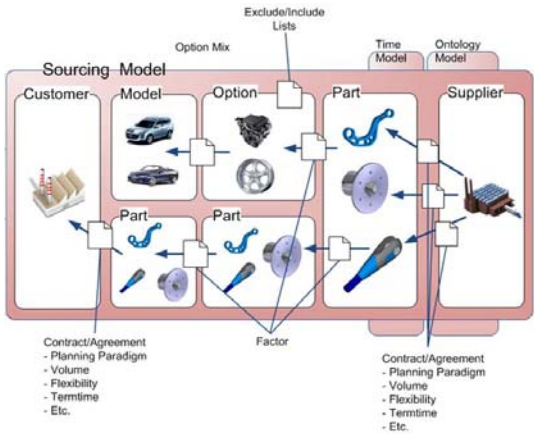


Fig. 1. Sourcing Model

made with each partner in the supply chain. Contracts encompass information on volumes and accepted deviations, term time, agreements on flexibility, and more. In order to create a connection from the customer to the supplier, products have to be split up in their constituent parts. The Bill of Materials (BOM) link contains factors for their calculation. There are two different ways to model BOM relations. First, there is the technical view, i.e. the afore mentioned conventional hierarchical decomposition. In addition to that, the definition of options is possible. Options combine several parts that in combination form a specific feature of the car, e.g. a sunroof. This can be used to express that some options cannot be combined with other options or that options require other options to be included.

5.2 Resource Model

The Resource Model contains all manufacturing resources that are relevant for planning (machines, workers, etc.) including their capacities. Regarding flexibility, this model also describes uncertainty concerning availability.

5.3 Adjustment Measure Model

The Adjustment Measure Model provides the necessary structures to represent network adaptivity. Network adaptivity also refers to local manufacturing and reflects in the division of measures into resource measures and network measures. They are identical except that the resource measures reference local resources whereas the network measures reference supply relationships. The rest are common attributes that describe measures by lead-time, effect, costs, availability, external approval, and pursued goal. The latter establishes a relationship to the overall goal model as it allows choosing that measure which supports the currently pursued goal best. The goal model is beyond the scope of this paper. Roughly, it is a decision support model for choosing alternative plans.

5.4 Demand Constraint Model

The Demand Constraint Model is the implementation of the entity's capacity and demand planning and realizes a central ILIPT requirement, viz. the direct connection of demand and capacity to allow real time capable-to-promise processes. It consists of three layers. Incoming demand is mapped to the Incoming Demand Layer where it is aggregated with existing demands. The Constraint Layer brings together demand and capacity. The Outgoing Demand Layer contains the demand that is forwarded to the suppliers. The model for coordination entities is set up a little different. It includes the Network Constraint Model and the Supply Demand Model. The former is like a Sourcing Model for the entire network and contains all supply relations between entities. No internal planning data is stored in this model. The latter is constructed in analogy to the Demand Constraint Model and maps the development of supply and demand on the network level.

5.5 Time Model

Companies employ calendars for their internal planning which are usually different from the Gregorian Calendar. Those calendars normally just number the workdays in ascending order starting at some arbitrary point in time. The Time Model provides structures for different calendars in order to create a common understanding of dates between customers and companies. It also introduces different planning horizons.

6 Application in a Multi-agent System

The models outlined in the previous section have been modeled as OWL ontologies using the editor Protégé. To make them available to the agents, source code had to be generated from the ontologies. Protégé makes use of the Eclipse Modeling Framework (EMF) for this. During code generation, each concept found in the ontology was translated to a corresponding Java interface. Each of the interfaces must now be implemented by a Java class. This means implementing the according interface and adding get and set methods as well as constructors. However, the semantic richness of ontologies cannot fully be translated to classes. This loss can be compensated by using a framework such as Jena to retain this knowledge for semantic checks. The resulting classes were then included in the agents which are constructed according to the BDI architecture.

Accordingly, the system is based on the Jadex platform. Each agent uses the models presented in this paper to manage its beliefs. Actual data comes from a MySQL database which currently serves as an ERP system simulation. Plans were programmed to give the agent control over its production planning. For the time being, no goals in the BDI sense were defined.

A proof of concept was designed which makes use of the models. The scenario encompasses an OEM producing cars and procuring parts from two suppliers.

Inter-enterprise planning is monitored by an instance called VOB according to ILIPT terminology. The planning process works as follows. First, the OEM receives an order and starts a requirement analysis. After performing a BOM explosion based on the data stored in the Sourcing Model, information on demands for each supplier is sent to the VOB. The VOB saves and forwards this information to the suppliers. Triggered by the incoming messages, the suppliers start capacity planning runs and check if the available workforce and machine time (and potentially other resources) are sufficient to comply with the request. This information can be accessed in the Resource Model. An answer is sent to the VOB which combines this with the OEM's demand in the Demand Constraint Model and sends an alert if a shortage is detected. In that case, the Adjustment Measure Model provides information on how the supplier's capacity can be increased at short notice. The Time Model is used to ensure consistency of dates.

The system can execute this process which is sufficiently backed by the presented data models. Further efforts have to be made to model bigger examples from practice to give substantial evidence for completeness or to detect shortcomings.

7 Summary and Outlook

In this paper, an ontology was presented that is able to model the domain in question, i.e. organizations in automotive supply chains, on a level of detail that allows for the use by IS, e.g. a multi-agent system. It was constructed bottom-up, starting from research on the domain and subsequently generating the abstract model, whereas previous works tried to come to applicable ontologies by iteratively refining a generic model. It was shown that this is a feasible way to come to models that are capable of being a semantic foundation for IS. The next step is to enhance the underlying communication processes by defining a fitting problem solving ontology. Then, the data formulation within planning entities can be detached from the general model and a comprehensive description of supply chain organizations and processes will be available for a certain field of application. We will use the resulting framework for further research, e.g. studies in supply chain cooperation.

References

1. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition 5(2), 199–220 (1993)
2. Guarino, N.: Formal Ontology and Information Systems. In: FOIS 1998, Trento, pp. 3–15. IOS Press, Amsterdam (1998)
3. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer, Berlin (2004)
4. Tsoukas, H., Vladimirou, E.: What is organizational knowledge. Journal of Management Studies 38(7), 973–993 (2001)

5. Chandra, C., Tumanyan, A.: Ontology Driven Knowledge Design and Development for Supply Chain Management. In: Proceedings 13th Annual Industrial Engineering Research Conference, Houston (May 2004)
6. Kuhn, A., Hellingrath, B.: Supply Chain Management. Springer, Berlin (2002)
7. Fayez, M., Rabelo, L., Mollaghasemi, M.: Ontologies for Supply Chain Simulation Modeling. In: Proceedings of the 2005 Winter Simulation Conference, pp. 2364–2370 (2005)
8. Chandra, C., Tumanyan, A.: Supply chain system taxonomy: development and application. In: Proceedings 12th Annual Industrial Engineering Research Conference, Portland (May 2003)
9. Chandra, C., Tumanyan, A.: Ontology-driven Information System for Supply Chain Management. In: Sharman, R., Kishore, R., Ramesh, R. (eds.) Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems, pp. 697–726. Springer, Berlin (2007)
10. Smirnov, A.V., Chandra, C.: Ontology-Based Knowledge Management for Cooperative Supply Chain Configuration. In: Proceedings of the American Association of Artificial Intelligence Spring Symposium, Palo Alto, pp. 85–92. AAAI Press, Menlo Park (2000)
11. Ye, Y., Yang, D., Jiang, Z., Tong, L.: An ontology-based architecture for implementing semantic integration of supply chain management. International Journal of Computer Integrated Manufacturing 21(1), 1–18 (2008)
12. Blomqvist, E., Levashova, T., Öhgren, A., Sandkuhl, K., Smirnov, A., Tarassov, V.: Configuration of Dynamic SME Supply Chains Based on Ontologies. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS, vol. 3593, pp. 246–256. Springer, Heidelberg (2005)
13. Ahmad, A., Mollaghasemi, M., Rabelo, L.: Ontologies for Supply Chain Management. In: Proceedings 13th Annual Industrial Engineering Research Conference, Houston (May 2004)
14. Umeda, S., Lee, T.: Management Data Specification for Supply Chain Integration, NISTIR 6703, National Institute of Standards and Technology, Gaithersburg (2001)
15. Fox, M.S.: The TOVE Project - Towards a Common-Sense Model of the Enterprise. In: Proceedings of the 5th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Springer, London (1992)
16. Grüninger, M., Atefi, K., Fox, M.S.: Ontologies to Support Process Integration in Enterprise Engineering. In: Computational & Mathematical Organization Theory 6, pp. 381–394. Kluwer Academic Publishers, Dordrecht (2001)
17. Parry, G., Graves, A. (eds.): Built To Order - The Road to the 5-Day Car. Springer, Berlin (2008)
18. OAGIS 9.0 - Open Applications Group Integration Specification, http://www.openapplications.org
19. Protégé, http://protege.stanford.edu

Semantic Extension of Agent-Based Control: The Packing Cell Case Study

Pavel Vrba[1], Miloslav Radakovič[1,2], Marek Obitko[1,2], and Vladimír Mařík[1,2]

[1] Rockwell Automation Research Center, Pekařská 695/10a,
155 00 Prague 5, Czech Republic
{pvrba,mradakovic,mobitko,vmarik}@ra.rockwell.com
[2] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27 Prague 6, Czech Republic

Abstract. The paper reports on the latest R&D activities in the field of agent-based manufacturing control systems. It is documented that this area becomes strongly influenced by the advancements of semantic technologies like the Web Ontology Language. The application of ontologies provides the agents with much more effective means for handling, exchanging and reasoning about the knowledge. The ontology dedicated for semantic description of orders, production processes and material handling tasks in discrete manufacturing domain has been developed. In addition, the framework for integration of this ontology in distributed, agent-based control solutions is given. The Manufacturing Agent Simulation Tool (MAST) is used as a base for pilot implementation of the ontology-powered multiagent control system; the packing cell environment is selected as a case study.

Keywords: manufacturing, distributed control, multi-agent systems, semantics, ontologies, semantic web.

1 Introduction

The theory of multi-agent systems (MAS) has been recognized as a promising paradigm for implementing the next generation of distributed, robust and dynamically reconfigurable automation control systems. The current trend of deployment of agents is obvious at all levels of the manufacturing business. At the lowest, real-time control level, so called holons or holonic agents are deployed. They are usually tightly coupled with the hardware as they directly interact with the real time control programs, implemented in standards such as IEC 61131-3 (e.g., ladder diagrams) or IEC 61499 (function blocks) [4]. Intelligent agents are also used for production planning and scheduling tasks both on the workshop and factory levels [18]. The application of MAS approaches is also relevant for implementing the vision of virtual enterprises, where various enterprises form temporary alliances for sharing their resources and competencies in order to increase their competitive advantage [5].

A multi-agent system is a federation of software agents interacting in a shared environment in order to cooperate and coordinate their actions given their own goals and plans [16]. The agent is designed to pursue its local objectives without direct external intervention, perceive and respond to changes in its environment and communicate with other agents in order to share information and cooperate on common goals. The strength of agent solutions is in the ability to exhibit complex behavior with the notions of global intelligence and self-organization even though the behaviors of individual agent instances could be quite simple.

In the past, the research in MAS field aimed at developing standards for designing, implementing and maintaining agent systems, with special attention paid to the inter-agent communication. The FIPA organization provided a comprehensive bundle of standards for multi-agent systems, including popular Agent Communication Language (FIPA-ACL) [10] and the Directory Facilitator (DF) for registering and searching of agent services. The ACL language defines an explicit structure of message including different slots like sender name, receiver name, language, content, etc. Compliance with the FIPA-ACL ensures syntactic interoperability of heterogeneous agent systems. The receiving agent is able to parse the slots of the message and even understand their meaning, i.e. who is the sender, what communication protocol this message follows, etc. However, such semantics is given just for the message level itself. When immersing down into the message content, the receiving agent might not be able to understand if it does not posses the same semantical description and interpretation of concepts in the message as the sending agent does.

With the evolution of semantic technologies in the past years the focus of researchers in the MAS area shifted naturally towards semantic interoperability and advanced methods for expressing and handling knowledge. Sharing information, goals and plans in the agent's social context as well as internal functionality of an agent in terms of maintaining knowledge and reasoning about it are more and more perceived as knowledge-intensive tasks.

The aim of this paper is to briefly review the state-of-the-art of applications of semantic techniques in MAS research related to manufacturing control systems. The proposal of generic manufacturing ontology and guidelines for its use in designing the agent-based control systems is discussed. The semantic extension of agent solution for packing line control is presented as a case study.

2 Ontologies for Distributed Manufacturing Control Applications

To facilitate knowledge sharing for the operation of the whole system and achieving common goals, agents must be able to understand information exchanged in messages. In order to have an extensible system, this understanding should be guaranteed even when the kind of communicated information is extended or changed, without any needs in reprogramming efforts. A possible way for ensuring the common understanding is the application of ontologies. Ontologies provide semantic description of a particular domain (such as manufacturing domain), i.e. they specify the meaning of entities and their relationships in the domain and define the parameters, constraints and possible consequences. Ontology is used by agents to represent the data and information about the environment as well as to share this data with other agents by means of messaging. In the context of Agent Communication Language (like mentioned FIPA ACL), the ontology is used for expressing the content of the message.

2.1 Ontologies in Manufacturing Systems

The ontologies for manufacturing systems have been discussed in detail for example in [17]. There are general purpose manufacturing ontologies such as MASON focusing on manufacturing operations [12], NIST's description of shop data model [15], definition of so called Automation Objects merging mechatronic and control models [13] or OOONEIDA focusing on the infrastructure of automation components [24]. The existing norms like ANSI/ISA-95 or ANSI/ISA-88 [3] should be considered as solid basis for creation of manufacturing ontologies. The ISA-95 „Enterprise-Control System Integration" standard defines hierarchical model of production organization and event flow and provides basic concepts for the integration of control system with business systems of the enterprise. The ISA-88 "Batch Control" describes in more detail the batch process production environment. It defines hierarchical model of production system from the enterprise, through areas and units down to control modules.

There are also ontologies created specifically for agent-based manufacturing systems – examples are ontologies for shop floor assembly [6], for reconfiguration purposes [1] or logistic planning [20]. These ontologies typically describe the layout of manufacturing equipment together with the description of what operations this equipment provides. Sometimes attention is paid also to low level control connection. The production is then described using sequence of operations, which may in addition specify tools needed to perform operations. Sometimes this sequence is described directly in ontology, not in the knowledge base. In general, the idea is to describe production equipment that is able to perform particular operations and use this description to find appropriate machines to perform production sequence.

We have studied these proposals but found that none of these ontologies provide a suitable semantic model that would fulfill requirements for generality, extensibility and deployability in agent-based discrete manufacturing control. The ANSI/ISA 88 is exclusively designed for batch processing industry what influences the way how physical model as well as process model is defined. The process model for instance defines process stages containing process operations that are furthermore composed of process actions. This seems to be too limiting in case that more than three levels are required in the hierarchy. In the proposed ontology presented in next section the production plan is composed of steps, where each step has either a basic operation or another complex plan associated. The same limitations and also some inconsistencies can be found also in the MASON ontology. Nevertheless, these ontologies provided inspiration for our ontology intended for flexible multi-agent manufacturing system.

2.2 Ontology for Flexible Manufacturing System

We have developed a set of ontologies in the Web Ontology Language (OWL) [7] (using Protégé editor) for description of customer orders, production plans, and the manufacturing system itself including transportation aspects. The goal of these ontologies is to serve as a base for further extensions for particular applications. For example, any new product type can be introduced with new types of parameters and new kinds of operations. The agents in the manufacturing system utilizing a shared ontology structure (see Sect. 3) do not have to be stopped and reprogrammed to absorb this

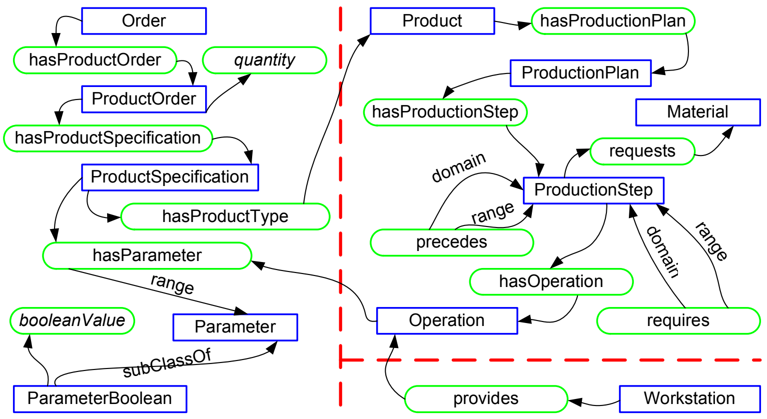


Fig. 1. Parts of ontologies for product orders, for production plans, and for production equipment; only selected OWL classes and relations are shown

change – it is only needed to extend ontologies (i.e., create subclasses of existing classes) to cover new products, new product parameters, new operations if needed, and to provide product plan describing the process of making the new product.

A segment of the proposed ontologies is shown in Fig. 1. The `Order` may consist of several `ProductOrders` that are specified by `ProductSpecification`. The specification of product parameters is expressed using subclasses of `Parameter` class, such as `ParameterBoolean` to specify Boolean values. These subclasses are usually product specific expressing for instance color, shape, material used, etc. Each `Product` has its own generic `ProductionPlan` expressed in a knowledge base, which is based on the extended production plan ontology.

The `ProductionPlan` consists of `ProductionSteps`. The sequence of `ProductionSteps` can be partially ordered by `precedes` relation. There is also possibility to specify required steps via `requires` relation, which is important for customization of the plan for a particular ordered product. Each `ProductionStep` has an `Operation` to be performed in this step together with possible `Parameters` that specify details of the `Operation`. The `Operation` is provided by `Workstation` that is a logical group of equipment (see next Sect.).

3 Integration of Manufacturing Ontology in Distributed Control System

This section provides general overview of the deployment of the proposed manufacturing ontology in distributed, agent-based manufacturing control applications. Each agent uses a specific piece of the proposed ontology for representing and processing the information related to the status and control of the manufacturing process. As well, the ontology is used for expressing the semantics of knowledge and data exchanged between the agents.

One of the basic elements of proposed framework is the concept of a *workstation*, considered as a logical composition of physical manufacturing devices and equipments, like machines, operators, buffers, etc. The workstation is represented by an autonomous control component – Workstation Agent (WA) – that can negotiate about the allocation of its advertised resources with the agents that control the execution of the production process. The single operation provided by a workstation is usually internally decomposed into the execution of several suboperations carried out by the particular equipment. Such an execution is supervised and controlled by the workstation agent by negotiations with the subordinate equipment agents. There are entry and exit points of the workstation where the transportation system is linked to and through which material, semi-products and products can be delivered into and out of the workstation.

Another key element in the proposed framework is the Product Agent (PA). In contrary to the usual comprehension of a product being a passive entity that never communicates or decides, the Product Agent implements the concept of an “active” or “intelligent” product. An active product is an intelligent entity associated with the physical product, which is able to perceive and act on the real state of the system [21]. It manages the information about itself and makes decisions relevant to its own life – it actively participates in negotiations with other agents about resource allocation and self-routing, it reports about its status and problems to the owner, etc. [16]. This definition, where one instance of PA corresponds to one instance of physical product contrasts to the definition of a *product holon* from the well known PROSA architecture. In PROSA the product holon contains the product model of the product type, not dealing with the state of a particular physical product instance being produced [22].

The other agents of the proposed architecture and their communication links are depicted in Fig. 2. The following sequence of steps takes place from receiving the order up to its fulfillment (numbering corresponds to labels of arrows in Fig. 2):

1. An order comes to the control system via a special Order system component, which in fact can also be an agent. It is supposed that the Order system has capabilities of receiving orders in different formats from different entities like human users through HMI, or from computer applications at the customer site sent over a computer network. The Order system converts the order parameters into an RDF format [14] using the OWL order ontology shown in Fig. 1.

2. The Order system creates a new instance of the Order Agent (OA) and passes it the RDF order specification. In this example, the order is composed of two product orders (see the association in Fig. 2), each for a particular type of product.

3. OA creates a new instance of PA for each product order. The PA is given only that part of the RDF order specification that corresponds to the product order. In this example, ProductAgent1 will take care of the fulfillment of ProductOrder1 and ProductAgent2 will process the ProductOrder2, respectively.

4. In order to obtain the instructions of how to make the product, the PA contacts the Production Plan Agent (PPA). This agent is supposed to manage and provide the instances of production plans for different product types implemented on the basis of general production plan ontology. Thus, this agent should not be confused with the PPA agent of the ExplanTech architecture [18] that schedules the production and thus corresponds rather to our PA agent. To select the proper plan, the PPA follows the `hasProductType` relation in the order specification and subsequently selects the plan associated with the `Product` by the `hasProductionPlan` relation (see Fig. 1).

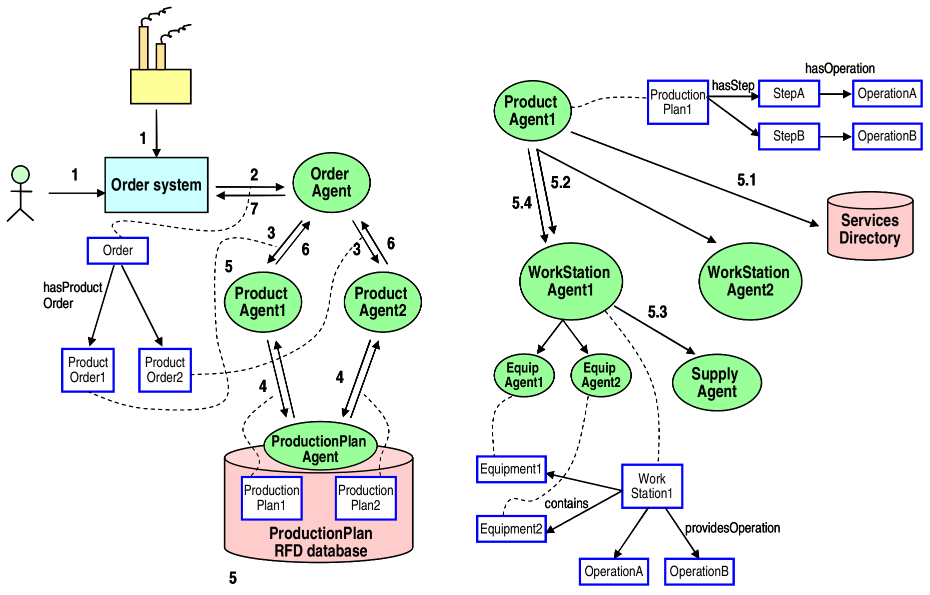


Fig. 2. Architecture for ontology-based multi-agent production control system

It is supposed that such a plan instance describes in general all the steps and operations needed to make the final product. However, such a plan has to be modified according to the order specification. This is basically done by PPA by assigning the parameter values from the order to parameters of operations in the plan or by deleting some production steps if for instance particular product feature is not required. Such a modified plan tailored to the specific product is returned back to the PA who can start to process it.

5. The PA processes the production plan ontology instance in terms of planning the execution of particular production steps (in the example depicted on Fig. 2 there are two steps: `stepA` and `stepB`). Planning of a single step consists of three main phases.

Planning phase. 5.1 The PA agent queries the Service Directory (usually the DF agent) to get the list of workstations that provide operation associated with the step. The association of the operation to step is through the `hasOperation` relation (see both Fig. 1 and 2). In this example, there is `operationA` associated with `stepA` and `operationB` associated with `stepB`, respectively.

5.2 The PA asks Workstation Agents to bid on the nearest available time slot when the operation could be done. The WA has to take into account the time needed to transport all required materials and semi-products from their current location to the workstation (see next step) as well as the time required for the operation itself.

5.3 The WA agent asks its Supply Agent (there is one instance of SA for each workstation) to plan the retrieval and transport of all material(s) and semi-product(s) required as an input to the production step. Some of the material can be already available in the workstation; some of them need to be transported from other work stations. The SA thus contacts all workstations providing materials asking for bidding on the time of delivery of required pieces of material.

Commit phase. 5.4 The PA sends request to such WA that offers the best bid (i.e., nearest time slot) for allocating previously offered resources. The WA subsequently requests SA to commit the retrieval and transport of materials.

Execution phase. In the last phase the PA asks the WA to start execution of the step. The execution phase for this step can be started right after all preceding steps are finished (Plan and Commit phases could be done in advance). The WA requests SA to execute the transport of all materials and semi-product(s). When all transport is done, the WA starts execution of the operation itself, by coordinating its subordinate agents. When the operation has finished (or even sooner), the PA starts planning the next step defined in the production plan. For this three-phase negotiation we have developed a Commit-Plan-Execute protocol, which is in detail described in [11].

6. When all the operations listed in the plan are done the PA informs the OA about finishing the product.

7. After the notification from all PAs the OA informs the Order system about fulfillment of the order.

The functionality inside the workstation can be advantageously described using the same concepts of the proposed ontology. Each of the workstation's externally advertised operation can have its own production plan instance associated. The workstation agent can obtain such plans from the PPA by asking it for all plans where the operations match those provided by the workstation's equipment.

4 DIAL Packing Cell Case Study

In this section we report on the latest developments related to the integration of ontologies in multi-agent manufacturing control applications. The platform that is currently being semantically extended is the Manufacturing Agent Simulation Tool (MAST). This agent-based automation control solution developed by Rockwell Automation provides means for design, simulation and real-world validation of a distributed control system based on intelligent agents [23]. Although named "intelligent", the agents contained in the MAST system can be viewed, in the semantic context of this paper, being rather very "simple". It is mainly because of the fact that semantic technologies were not as enhanced in a year 2000 when the development of MAST began as they are today. There were FIPA standards, newly emergent agent platforms like JADE and FIPA-OS and the XML as the latest technological advancements available those days. So, the communication of agents is based on XML messages, which semantic interpretation is not described explicitly, but is tightly embedded in the agent program code.

Recently, we have reimplemeted the MAST agents in such a way that the communication, representation of knowledge and reasoning is based on ontologies. Additionally, we have provided a flexible framework for explicitly defined behavioral modules [19], which can be dynamically modified in the agent (agent can receive a new behavior module in binary form via message) as the ontology is being extended or updated (see more in Sect. 5). The main objective for this new, semantic-based realization is the creation of a semantical description of the production process using the manufacturing ontologies described in Sect. 2., and integration of the support for ontologies in agents taking part in production planning and control.

4.1 The DIAL Packing Environment and Non-semantic Agent Control Solution

The packing and assembly environment installed in DIAL (Distributed Information and Automation Laboratory), formerly known as CDAC, at Cambridge University's Institute for Manufacturing [8] has been previously used as a real-world platform for verification of the MAST agent-based control principles [9]. The prototype production includes packing of gift boxes that contain arbitrary combination of three cosmetic items, which the user specifies in his/her order by selecting from four different types (gel, razor, deodorant and foam). These items are stored in a vertical, four-slot storage unit, from where they are picked up by a Fanuc M6i robot. The line contains two of such independent production stations (marked as Workstation1 and Workstation2 in Fig. 3) and one Automated Storage and Retrieval System (ASRS) composed of a portal crane and three rack storages (marked as Workstation3 in Fig. 3). The transportation between the workstations is provided by a monorail conveying system with shuttles carrying boxes. As can be seen in Fig. 3 there is one main horizontal transportation loop servicing ASRS and two vertical loops for reaching the production stations.

For the agent-based simulation of the DIAL testbed the MAST system has been extended with a set of new agents controlling the behavior of specific equipment. The storage unit agent maintains the list of available raw items, the Fanuc robot agent controls pick-and-place operations of the robot, the rack storage agent provides empty boxes and the gantry robot agent controls the movement of boxes by the portal crane. The task of the order agent, automatically created with each new order, is to find a suitable product agent that will actually control the production process. For simplicity of implementation, the already available shuttle agent controlling the movement of a shuttle has been extended by the product agent-like behavior. This agent negotiates about elementary operations with individual resource agents. First, it negotiates with rack storage agents to obtain an empty box and subsequently with the gantry robot agent to pick the box out of rack and place it on the shuttle. Then, for each item to be packed it negotiates with the storage units to find the one that can supply the required item and subsequently, after shuttle with box reaches the production station, asks robot to pack the item to the given slot in box. When all slots are filled the finished product returns to ASRS what involves negotiation with rack storage agents again to provide an empty place for final storage. The negotiations are based on the contract-net protocol in which the list of participants requested to bid on specific operation is obtained from the Directory Facilitator that manages the registration of available agent services.

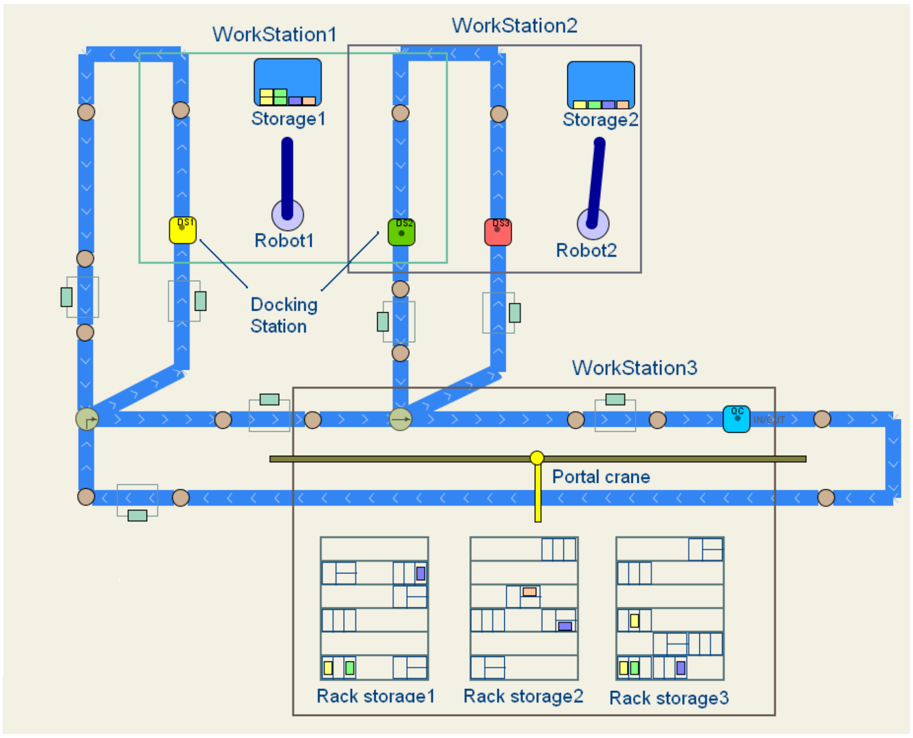


Fig. 3. Prototype packing line of University of Cambridge's DIAL lab simulated in MAST system

4.2 Ontology-Based Implementation of DIAL Scenario

To demonstrate the usability of the developed manufacturing ontology presented in Sect. 2, the DIAL scenario-specific extension has been created on top of the general concepts provided by the ontology. There is a new `PackedBox` class extending the general `Product` class that is used to select corresponding production plan. Additionally, there is `BoxItemType` class representing the type of item inserted into the box with subclasses `Deodorant`, `Gel`, `Razor` and `Foam`. These subclasses are used in the specification of order as parameters specifying what type of item is requested for each slot in a box. Furthermore, the production plan for making the gift box has been created – it is illustrated in Fig. 4. It is composed of five steps; the first one is for retrieving an empty box from ASRS (associated with `BoxRetrieving` type of operation); next three steps, which can be executed in any order, concerning inserting items into particular slots in the box (operation `BoxItemPacking`) and in the last step the finished box is stored back into the ASRS (operation `BoxStoring`).

The orders are placed through the OrderManager component in the graphical user interface. Its task is to compose a semantic description of the order based on parameters specified by the user in the form of an RDF description compatible with the order ontology. The order agent created for the order then instantiates new product agent.

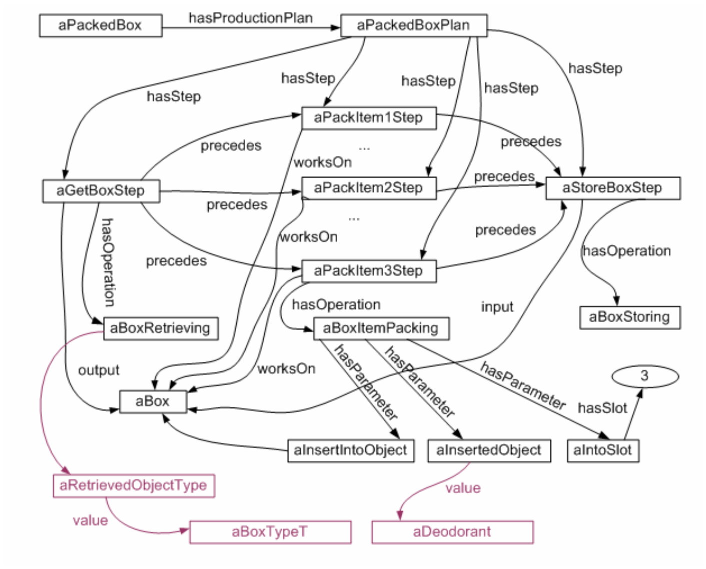


Fig. 4. Ontology describing production process (box packing) in DIAL scenario. At the bottom there are schematically shown parameters (aBoxTypeT and aDeodorant) copied from user order when Production Plan Agent prepares tailored production process description.

The product agent receives the order specification from the order agent. The product agent then asks a special Production Plan Agent for an instance of production plan ontology for box packing. This agent modifies the general production plan according to the user's order. This modification includes association of parameters from the order to particular production steps (type of box and types of inserted items – can be seen at the bottom of Fig. 4). Optionally, deletion of some of the three packing steps is done in case particular slot is requested to be empty.

According to this production description, the product agent plans execution of these steps, i.e. searches the Directory Facilitator for agents providing operations associated with steps (`BoxRetrieving`, `BoxItemPacking` and `BoxStoring`). The cost-based model, extended contract-net protocol [11] and RDF content language are used for negotiation about resource allocation between the product agent and the workstation agents. In bidding on cost of providing operation the workstation agent considers the availability of raw material (boxes and cosmetic items), capacity and load of workstation and product priority. The product agent selects the lowest cost and subsequently negotiates with the transporter agents representing shuttles about the delivery of product to the selected destination. Once the operation is finished the product agent starts planning of the next step defined in the plan ontology.

5 Key Characteristics and Advantages of Combining Agents with Semantics

The shift from non-semantic multi-agent control systems to the next generation of semantically enriched solutions incorporating ontologies for representing and sharing knowledge can be compared to the transition from classical centralized control to distributed agent-based systems. This section summarizes some key aspects of combining agents with semantic technologies and discusses potentials and advantages of such an integration.

Duality of knowledge. It has been discussed that the ontology provides a semantic description of the particular domain knowledge, like for instance manufacturing. It defines basic concepts or facts, like machines and operations and captures relations between those concepts like for instance capturing the fact that a machine provides an operation. The ontology thus provides a *declarative* description of the knowledge. It is used by an agent to give the semantic meaning to the data the agent obtains by perceiving the environment or by receiving from other agents via messages. The knowledge about some domain has to be however considered as twofold. Especially in the applications where the agents act in a real environment, like they are controlling machines in factories, the agent has to know not only *what does it mean* but also *what to do* when a new observation is done. In other words there might be an action required to be taken as a response to a new information in agent's repository. In fact, this is also a king of knowledge, although represented by an algorithm, procedure or process to be executed. Thus we call this *procedural knowledge* and argue it is necessary to interlink it with the declarative one. This dual character of knowledge is not discussed so much in the literature. Rzevski et al. in [20] present the Magenta agent toolkit where the ontology provides also an associated procedural description of the agent behavior.

Modularity. The usual concept of inheritance coming from object oriented programming is deployed when building the ontologies. It means that the existing classes representing general entities are used to create subclasses describing more specialized entities (like drilling being a subclass of an operation). This concept is naturally used to build a hierarchy of domain or process specific knowledge modules, describing for instance production planning and scheduling aspects, material routing, ordering, machining, etc. A key point is that an agent, designed for a particular task, does not need to be equipped with the overall ontology; it gets only an appropriate set of knowledge modules that are needed to describe the agent's domain of expertise. Complementary to the a tailored subset of ontology, i.e., the declarative knowledge, the agent should also be provided with a corresponding subset of procedural knowledge. This mechanism allows creating minimal, tailored agents in a modular way, having only necessary amount of knowledge and functionality. This aspect is important for instance when considering deploying agents on embedded devices or controllers with strict memory and computing power constraints.

We have developed a pilot implementation of these principles in MAST agents. As presented in more detail in [19], the procedural knowledge is captured in the modules of behaviors programmed and stored in separate Java classes held apart from the main agent body. When the agent is instantiated it is given the tailored subset of these modules needed for its common functionality. When the agent notes an unusual event for which it does not have appropriate behavior it can obtain it on-line from dedicated Behavior Repository agent(s). This agent manages the database of behavior modules for different types of agents and on request sends these behaviors to the agents in a binary form. This mechanism can be also used for sending updates of existing behaviors without a need to stop the running agents and reprogram them off-line. The Magenta toolkit presented in [2] also allows sending updates of agent's behavior (program code) at runtime as the ontology is being modified or extended.

We envision that eventually the agents will not be programmed like today, where there are specific agent classes representing specific manufacturing components or tasks like conveyor belt agent, diverter agent or product agent. Instead, there will be one agent class representing a general agent skeleton. When instantiated, the agent downloads, based on its type, all the needed ontology as well as behavior modules.

Flexibility. Even if we will not consider modifying or updating agent program code at runtime, the use of ontologies increases the flexibility of the agent-based control system. In the packing cell scenario described in Section 4, the previously implemented product agent controlling the packing of box had its behavior fixed to this particular product type. Although it could discover new operation providers added to the system or find alternative routing in case of conveyor failures, the same product agent class could not be used for controlling different type of production process, like for instance box unpacking. In the new, semantically enriched solution, the production process is not described in the program code of the agent, but it is specified explicitly using the ontology. The agent is then able to process such a recipe automatically, in such a way it schedules the execution of steps by negotiating with workstation agents about providing the operations associated with the steps. The great advantage is that the same product agent can be simply used to control any kind of production process, if it is described in the same ontology. In the packing cell scenario the plan is to develop the RDF description of the box unpacking process and use the existing product agent to fulfill the order for unpacking a box.

6 Conclusions

The main objective of the semantic extension of the agent-based control that is discussed in this paper is to design a coherent framework providing general and extendable ontologies for semantic description of manufacturing tasks. In addition, the instructions for integrating these ontologies in agent solutions for discrete production control are provided. The agent simulation system MAST together with the DIAL packing line has been selected as platforms for verification of the developed framework. Compared to previous non-semantic solutions the major advantage of the new ontology-based system is the introduction of a general product agent that does not have behavior fixed to any concrete product type. It is able to control production of any product type following the semantic description of the production plan. In future work, the focus will be on implementation of a general workstation agent (currently it is still fixed to particular equipment) that is able to govern the functionality of arbitrary aggregation of equipment. It also uses the production plan ontologies defining how the externally advertised operation breaks down into elementary actions executed by the equipment. Other plans include using ontologies and rules for basic reasoning of agents instead of pure Java code, for example for matchmaking, for reasoning in transportation or for generation of tailored production plan from general one based on customer order parameters.

Acknowledgements

This research has been supported by Rockwell Automation, by the Ministry of Education of the Czech Republic within the Research Programme No.MSM6840770038: Decision Making and Control for Manufacturing III and by the FIT-IT: Semantic Systems program, an initiative of the Austrian Federal Ministry of Transport, Innovation, and Technology (bm:vit) under contract FFG 815132. The authors would like to thank to Prof. Duncan McFarlane, the head of the Distributed Information & Automation Laboratory within the Institute for Manufacturing, University of Cambridge for his support and cooperation.

References

1. Al-Safi, Y., Vyatkin, V.: An Ontology-Based Reconfiguration Agent for Intelligent Mechatronic Systems. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 114–126. Springer, Heidelberg (2007)
2. Andreev, V., Rzevski, G., Skobelev, P., Shveykin, P.: Adaptive Planning for Supply Chain Networks. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 215–224. Springer, Heidelberg (2007)
3. ANSI/ISA-88.01.1995. Batch Control Part 1: Models and Terminology. American National Standard. The Instrumentation, Systems and Automation Society (1995)
4. Brennan, R., Vrba, P., Tichy, P., Zoitl, A., Sünder, C., Strasser, T., Mařík, V.: Developments in Dynamic and Intelligent Reconfiguration of Industrial Automation. Computers in Industry 59/6, 533–547 (2008)
5. Camarinha-Matos, L.M.: Multi-Agent Systems In Virtual Enterprises. In: Proceedings of International Conference on AI, Simulation and Planning in High Autonomy Systems, pp. 27–36. SCS publication, Lisbon (2002)
6. Cândido, G., Barata, J.: A Mutliagent Control System for Shop Floor Assembly. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 293–302. Springer, Heidelberg (2007)
7. Dean, M., Schreiber, G.: OWL Web Ontology Language reference (2004), `http://www.w3.org/TR/owl-ref/`
8. Fletcher, M., Brusey, J.: The Story of the Holonic Packing Cell. In: Proceedings of 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems. ACM Press, Melbourne (2003)
9. Fletcher, M., Vrba, P.: A Brace of Agent Simulation Scenarios. In: Proceedings of IEEE Workshop on Distributed Intelligent Systems, pp. 169–176. Prague (2006)
10. Foundation For Intelligent Physical Agents: FIPA ACL Message Structure Specification, SC00061G (2002)
11. Kadera, P., Tichy, P.: Plan, Commit, Execute Protocol in Multi-agent Systems. In: HoloMAS 2009, Linz (submitted, 2009)
12. Lemaignan, S., Siadat, A., Dantan, J.-Y., Semenenko, A.: MASON: A proposal for an ontology of manufacturing domain. In: IEEE Workshop on Distributed Intelligent Systems, pp. 195–200. IEEE Computer Society Press, Los Alamitos (2006)
13. Lopez, O., Martinez Lastra, J.L.: Using Semantic Web Technologies to Describe Automation Objects. International Journal of Manufacturing Research 1(4), 482–503 (2006)
14. Manola, F., Miller, E.: RDF primer (2004), `http://www.w3.org/TR/rdf-primer/`

15. McLean, C., Lee, Y., Shao, G., Riddick, F.: Shop Data Model and Interface Specification, NISTIR 7198 (2005)
16. Meyer, G., Främling, K., Holmström, J.: Intelligent Products: A survey. Computers In Industry 30(3), 137–148 (2009)
17. Obitko, M., Vrba, P., Mařík, V., Radakovič, M.: Semantics in Industrial Distributed Systems. In: Proceedings of the 17th IFAC World Congress, pp. 13880–13887 (2008)
18. Pěchouček, M., Rehák, M., Charvát, P., Vlček, T., Kolář, M.: Agent-Based Approach to Mass-Oriented Production Planning: Case Study. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 37(3), 386–395 (2007)
19. Radakovič, M., Vrba, P., Obitko, M.: Architecture for Explicit Specification of Agent Behavior. In: 13th IFAC Symposium on Information Control Problems in Manufacturing, Moscow, Russia (accepted 2009)
20. Rzevski, G., Skobelev, P., Andreev, V.: MagentaToolkit: A Set of Multi-agent Tools for Developing Adaptive Real-Time Applications. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS, vol. 4659, pp. 303–313. Springer, Heidelberg (2007)
21. Sallez, Y., Berger, T., Tretensaux, D.: A Stigmergic Approach for Dynamic Routing of Active Products in FMS. Computers in Industry 30(3), 204–216 (2009)
22. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeter, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers in Industry 37(3), 255–274 (1998)
23. Vrba, P., Mařík, V.: Simulation in Agent-Based Control Systems: MAST Case Study. In: Proceedings of 16th IFAC World Congress, Prague (2005)
24. Vyatkin, V., Christensen, J., Lastra, J.: OOONEIDA: An Open, Object-Oriented Knowledge Economy for Intelligent Industrial Automation. IEEE Transactions on Industrial Informatics 1(1), 4–17 (2005)

Product Design Network Self-contextualization: Enterprise Knowledge-Based Approach and Agent-Based Technological Framework

Tatiana Levashova[2], Kurt Sandkuhl[1], Nikolay Shilov[2], Alexander Smirnov[2], and Vladimir Tarasov[1]

[1] School of Engineering at Jönköping University
Box 1026, 55111 Jönköping, Sweden
{Kurt.Sandkuhl,Vladimir.Tarasov}@jth.hj.se
[2] St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, 39, 14 line, St.Petersburg, 199178, Russia,
{oleg,smir,nick}@iias.spb.su

Abstract. The paper introduces self-contextualization in a service infrastructure for product design networks as novel application field for multi-agent technology. The main contributions of this paper are (1) identification of requirements from product design networks to the supporting service infrastructure, (2) the use of enterprise knowledge modelling techniques for the representation of computable context models, (3) a technological framework based on agent technology for self-contextualization based on enterprise knowledge models.

Keywords: self-contextualization, multi-agent system, enterprise model, engineering design network.

1 Introduction

The objective of this paper is to introduce and describe an application area for multi-agent technology within collaborative engineering for networked manufacturing enterprises, which so far has not received much attention in research: the self-contextualization of an agent-based service infrastructure in product design networks. Collaborative engineering[1] in general is computer-supported design, development or production planning work in dispersed groups that is based on a joint technical infrastructure and common collaboration objectives. Product design is a sub-area of collaborative engineering, which usually involves different engineering disciplines, stakeholders from various departments in an enterprise and external suppliers or collaborators. In many industrial domains, shorter innovation cycles lead to a larger number of parallel design projects with changing team members.

In order to efficiently support product design, networks have to be flexible. In other words, they have to quickly readjust relationships between their members responding to changes in the environment (e.g., market needs and requirements, appearance of new technologies). This presents a number of problems. Among the options

[1] [3] describes selected technologies and applications of collaborative engineering.

Table 1. Business conditions favouring major strategies for product design network value maximization

Strategy	**Current Situation Uncertainty**	**Flexibility**	**Cost of Lost Sale**	**Cost of Postponement**
Postponement	Average-High	Low	High	Low
Information Sharing	Average-Low	High	Low	High

for maximizing a value from design networks, there are two major strategies: (i) *postponement* (delayed differentiation until customer's demand for specific end products) and (ii) *information sharing* (for faster and more accurate information flow across the supply network). Certain business conditions that favour each strategy are depicted Table 1 [1]. In this paper, the information sharing strategy is considered as is more appropriate for engineering design networks.

Knowledge sharing and exchange in a flexible product design network are highly important and should be achieved at both technical and semantic levels. The interoperability at the technical level is addressed in a number of research efforts. It is usually represented by such approaches as e.g., SOA (service-oriented architecture) [9]. The semantic level of interoperability in the flexible product design network is also paid significant attention. As an example (probably the most widely known), the Semantic Web initiative is worth mentioning. The main idea is to use ontologies for knowledge and terminology description. Examples for knowledge sharing and reusable components based on ontologies are presented in [10, 11, 12].

The approach presented in this paper also relies on the ontological knowledge representation for its sharing. Ontologies facilitate information retrieval over collections of distributed and heterogeneous information sources; they help to provide for semantic integration of information and facilitate interoperability between heterogeneous knowledge sources at high level of abstraction. The ontology describes common entities of the enterprise knowledge model and relationships between them. As a result, it is possible to treat all available knowledge and competencies as one distributed knowledge base.

Furthermore, the dynamic nature of flexible product design networks requires considering the actual situation in order to provide for up-to-date knowledge or information. For this purpose, the idea of contexts is used. Context represents additional information that helps to identify specifics of the current transaction. It defines a narrow domain that a particular person is working with.

The main contributions of this paper to the research area are (1) identification of requirements from product design networks to the supporting service infrastructure, (2) the use of enterprise knowledge modeling techniques for the representation of computable context models, (3) a technological framework based on agent technology for self-contextualization based on enterprise knowledge models. The next section will present requirements of product design networks based on an industrial scenario and introduce the overall approach for self-contextualization. Section 3 presents the enterprise knowledge modelling aspects of the approach; section 4 the technological framework. Summary and conclusions are given in section 5.

2 Industrial Requirements and Overall Approach

This section introduces the service infrastructure requirements from product design networks based on an industrial case and outlines the approach for self-contextualization of the services meeting these requirements**.**

2.1 Industrial Case

The industrial case, which formed the context for work presented in this paper, is taken from automotive industries. The case focuses on distributed product development and multi-project lifecycles in a networked organization with different suppliers. During the EU-project MAPPER, analysis of requirements for collaborative engineering support, development of a collaboration infrastructure and application of this infrastructure in everyday work was performed in the industrial case [4].

The main partner in the case is the business area "seat comfort components" of a first tier automotive supplier with the main product development sites in Scandinavia. The seat comfort products mainly include seat heater, lumber support, seat ventilation, climate control and head restraint. The focus was on the advanced engineering unit, where development tasks are concentrating on new concepts and new materials. This includes elicitation of system requirements based on customer requirements, development of functional specifications, design of technical architecture, co-design of material, electrical and mechanical parts, integration testing and production planning.

The process is geographically distributed involving engineers and specialists at several locations of the automotive supplier and sub-supplier for specific tasks. A large percentage of seat comfort components can be considered as product families, i.e. various versions of the components exist and have to be maintained and further developed for different product models and different customers. In this context, flexible product development in networks with changing partners on customer and sub-supplier side is of crucial importance.

The main challenges for collaborative engineering are to support geographical distribution and flexible integration of changing partners, to enable flexible engineering processes reflecting the dynamics of changing customer requirements and at the same time support well-defined processes for coordinated product development, to coordinate a large number of parallel product development activities competing for the same resources, and to allow for richness of variants and at the same time reuse and generalization of products. The overall target for the product development phases within the automotive business is to enhance the quality and reduce the time to market for new products with the lowest price and weight of the product. Collaboration between people within the company and with external partners is a key success factor to meet these basic needs.

The MAPPER project performed a requirements analysis in the above use case focusing on four types of requirements [5]:

- *Approach* requirements, relevant in cases where several alternative approaches exist for an activity in the use case, e.g. line vs. matrix organization of projects.
- *Methodology* requirements include any requirement related to methodology use and design.

- *Platform* requirements include requirements with respect to the collaboration infrastructure or information resources (document templates, model templates, instructions, guidelines, etc.).
- *Solution* requirements concern implementation characteristics of the solution for a specific project.

In the context of this paper, solution and platform requirements are of particular interest. Selected requirements in these two areas include:

- Integration of existing tools and services at the different partner organisations, like ERP or PLM systems, repositories, document management systems or information services supporting execution and ad-hoc changes in enterprise knowledge models (see section 4) for workflow and methodology support of collaborative engineering
- Remote invocation of tools available at other sites or other partners in the distributed design network (e.g. expensive simulation tools or services)
- Task management for the different roles and stakeholders in the dispersed team
- Support for group interaction including synchronous services (e.g. online conferencing) and asynchronous services (e.g. group repository)

The above requirements confirm the need for a dynamic support of collaborative engineering based on enterprise knowledge with focus on knowledge sharing.

2.2 Conceptual Approach

The main elements of the proposed approach are enterprise knowledge models for capturing the required information for self-contextualisation at design time and an agent-based framework realising the self-contextualization at runtime. This section provides an overview, both elements will be introduced in more detail in sections 3 and 4.

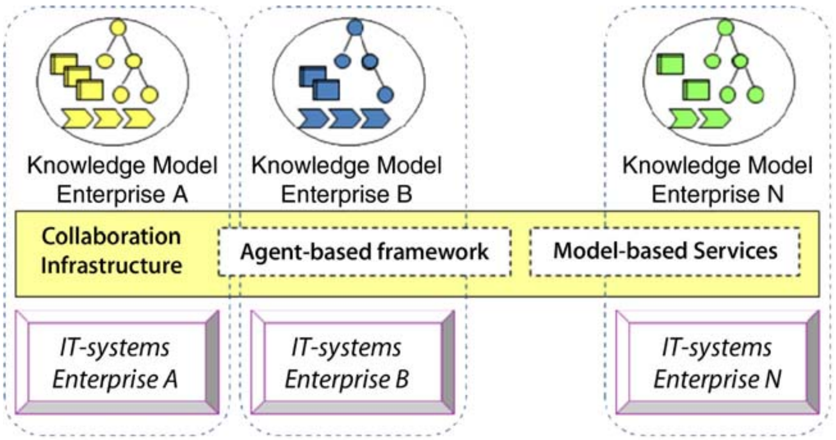

Knowledge Model Enterprise A
Knowledge Model Enterprise B
Knowledge Model Enterprise N
Collaboration Infrastructure
Agent-based framework
Model-based Services
IT-systems Enterprise A
IT-systems Enterprise B
IT-systems Enterprise N

Fig. 1. Main elements of the overall approach for self-contextualization

Figure 1 illustrates the overall approach for self-contextualization proposed in the paper. The relevant enterprise knowledge from the different partners contributing to the distributed product design is captured in enterprise knowledge models, which could be represented as enterprise ontologies or enterprise models. These knowledge models include specifications of the relevant work processes, product structures, organizational roles involved and IT-systems and services used. The models are executable, for example in workflow engines or the collaboration infrastructures like the MAPPER infrastructure [4]. At the same time, these models serve as input for self-contextualization of services supporting collaborative product design. Self-contextualizing services are expected to complement the knowledge about the product design work captured at design time in the enterprise knowledge models with knowledge about the current status of available collaboration infrastructure and network partners at runtime. An example could be simulation services for investigating specific product characteristics and performance measures. The general need for such a service and the type of simulator would be specified in the enterprise knowledge model at design time, but the configuration of the simulator, collection of input parameters from and distribution of results to the participating partners depends on the context at runtime.

Figure 2 illustrates principles and technologies for enabling self-contextualization of the agent-based services in the product design network.

In the approach the resource functionalities are modelled by Web-services. This makes it possible to replace the self-organization of the resources with that between the appropriate Web-services. For Web-service interactions formal interface agreement defined by the technology of Web-services is used. To provide the Web-services

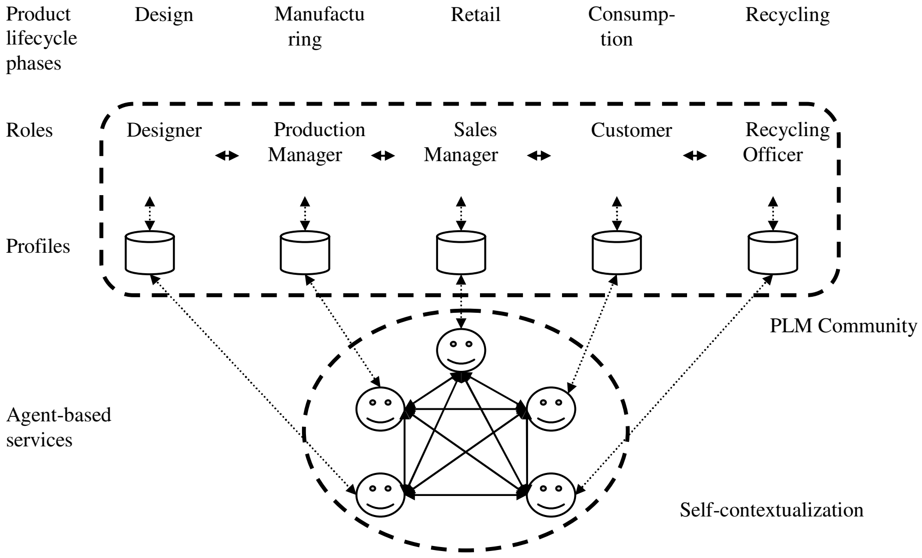


Fig. 2. Reference Model of Product Design Network Self-Contextualization

with semantics the Web-service descriptions are aligned against an ontology. The paper extends the presented earlier research [2] introducing self-organisation into the developed context-based approach to coalition operations.

3 Enterprise Knowledge Modeling

Enterprise knowledge modeling in general is applying and extending approaches, concepts and technologies from enterprise modeling and ontology engineering for knowledge representation and knowledge-based solutions. Enterprise modeling is an established research field, which is influenced by Enterprise Engineering, a field commonly considered as part of Industrial Engineering, and (Business) Process Engineering usually considered as part of information systems and informatics. In general terms, enterprise modeling is addressing the systematic analysis and modeling of processes, organization structures, products structures, IT-systems or any other perspective relevant for the modeling purpose. Enterprise models can be applied for various reasons, like visualization of current processes and structures in an enterprise, process improvement and optimization, introduction of new IT solutions or analysis purposes.

Within the MAPPER project, collaborative engineering was supported by enterprise knowledge models capturing best practices for reoccurring tasks in networked enterprises. These best practices were represented as active knowledge models using the POPS* perspectives. Active knowledge models are visual models of selected aspects of an enterprise, which cannot only be viewed and analyzed, but also executed and adapted during execution. The POPS* perspectives include the enterprise's processes (P), the organization structure (O), the product developed (P), the IT system used (S) and other aspects deemed relevant when modelling (*)[7].

The term "task patterns" was introduced for these adaptable visual models, as they are not only applicable in a specific company, but are also considered relevant for other enterprises in automotive supplier industry. Task pattern in this context is defined as "self-contained model template with well-defined connectors to application environments capturing knowledge about best practices for a clearly defined task" [8].

Figure 3 shows an example enterprise knowledge model. This example visualizes in the upper part the process perspective of the task pattern. The process flow with the steps "1. Prepare draft", "2. Material Testing", "3. Process Trial" and "4. Release Material Specification" is shown. For all these process steps, various refinement levels are part of the task pattern, which could not be shown in the figure due to complexity reasons. Above the process flow, objectives and documents which are input to the task pattern are included. The arrows indicate relationships between processes, roles, systems and documents or objectives. On the lower part, the roles involved in the process are included (grouped at the left hand side) and the IT systems and tools are shown.

Modeling such product development knowledge in the industrial case was performed according to the C3S3P methodology. C3S3P is based on work in EU projects from the area of networked enterprises. In order to support solutions development for such extended enterprises, the EXTERNAL project developed a methodology for extended enterprise modelling [6], which initially was named SGAMSIDOER. This

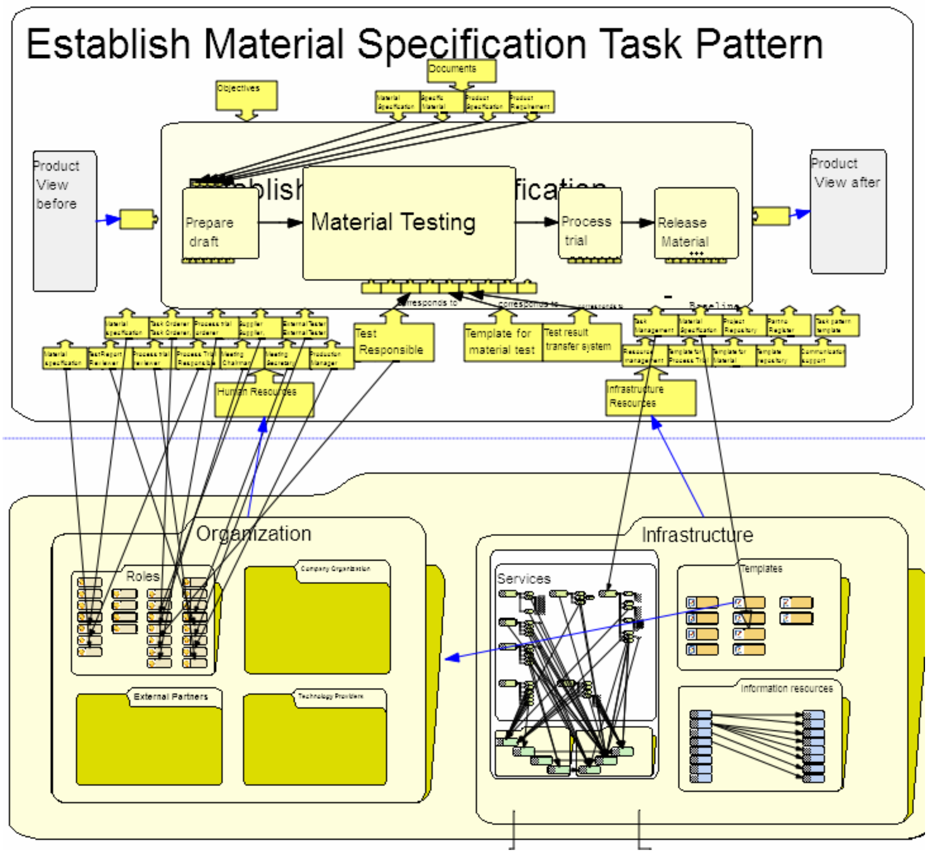


Fig. 3. Task pattern “Establish Material Specification”

methodology was further developed towards a complete customer delivery process denoted C3S3P, which was used in the ATHENA and MAPPER projects. C3S3P distinguishes between seven stages called Concept-study, Scaffolding, Scoping, Solutions-modelling, Platform integration, Piloting in real projects and Performance monitoring and management.

4 Agent-Based Framework for Self-contextualization

The section introduces the framework for self-contextualization in two parts: the technological framework (4.1) and the service-base architecture (4.2).

4.1 Technological Framework

The overall technological framework is shown in Table 2. The approach focuses on three types of resources to be organized: *information*, *problem-solving*, and *acting*. *Information* resources are resources providing information from information sources as sensors, Web-sites, databases, radio-frequency identification tags, etc. *Problem-solving* resources are computational modules, applications, services, etc. that can be used to solve problems requiring solutions in the current situation. *Acting* resources are organizations and persons taking joint actions in the current situation according to their roles.

To model the functionalities of the resources the technology of Web-services is applied. The functions (services) provided by the resources are modeled by a set of Web-services so that each service provided by a resource is modeled by one Web-service.

To the resources would be able to exchange information about their needs and capabilities and to share their knowledge, they are supported by a central application ontology (AO). The AO represents knowledge of the logistics domain. It formalizes conceptual knowledge of the domain and specifies problems that may require solutions in situations caused by different types of logistic related tasks. In the AO problems are represented as classes, problem arguments are represented as class properties. The AO does not hold instances. The ontology classes are instantiated in a particular situation (context) by the resources of the smart environment.

Agreement between the resources and the AO is expressed through alignment of the descriptions of the Web-services modeling the resource functionalities and the AO. This makes the Web-services provided with semantics. The operation of the alignment is supported by a tool that identifies semantically similar words in the Web-service descriptions and the AO.

A situation is modeled at two levels: abstract and operational. The levels are represented by abstract context and operational context, respectively. The abstract context is an ontology-based situation model embedding the specification of problems to be solved in this situation. It is created by core Web-services incorporated in the environment. These Web-services are considered as environment components.

The operational context is an instantiated abstract context and the real-time picture of the logistic network. Producing the operational context is one of the purposes of

Table 2. Technological Framework

Tasks Solved	*Solution*	*Technology*	*Result*
Application ontology creation	Integration of existing ontologies	Ontology engineering Ontology management	Application ontology
Information resource representation	Alignment of the descriptions of the resources and the application ontology	Ontology engineering Ontology management Web-Services	Application Ontology with references to information resources
Knowledge acquisition and organization	Abstract context creation	Ontology management	Task model (abstract context)
Information acquisition and organization	Operational context creation	Context management Web-services	Instantiated task model (operational context)
Solution search	Self-organization	Intelligent agents Web-services	Solution

resource self-organization. On the grounds of the resources are represented by sets of Web-services, the self-organization of the resources is replaced with that between the appropriate Web-services. Besides the operational context producing, the Web-services are purposed to solve problems specified in the abstract context and to get acting resources to take part in the activities.

4.2 Service-Based Architecture

In the architecture (figure 4) of the system intended for functioning in the smart environment two types of Web-services are distinguished: *core Web-services* and *operational Web-services*.

The core Web-services are intended to creation of the abstract context and to monitoring the environment. The core Web-services comprise:

- **MonitoringService** monitors the smart environment, identifies the type of the logistic task, and produces a corresponding message.
- **AOAccessService** provides access to the AO;
- **AbstractContextService** creates, stores, and reuses abstract contexts;
- **ManagementService** manages Web-services to create the abstract context. It operates with the service registry where the core services are registered.

The operational Web-services self-organize a network. To make the Web-services active components capable to self-organise an agent-based service model is used. Agents are intended to negotiate services' needs and possibilities in terms of the AO and "activate" Web-services when required. The services' needs and possibilities are respectively input and output arguments of the functions that the Web-services implement. The set of operational Web-services comprises:

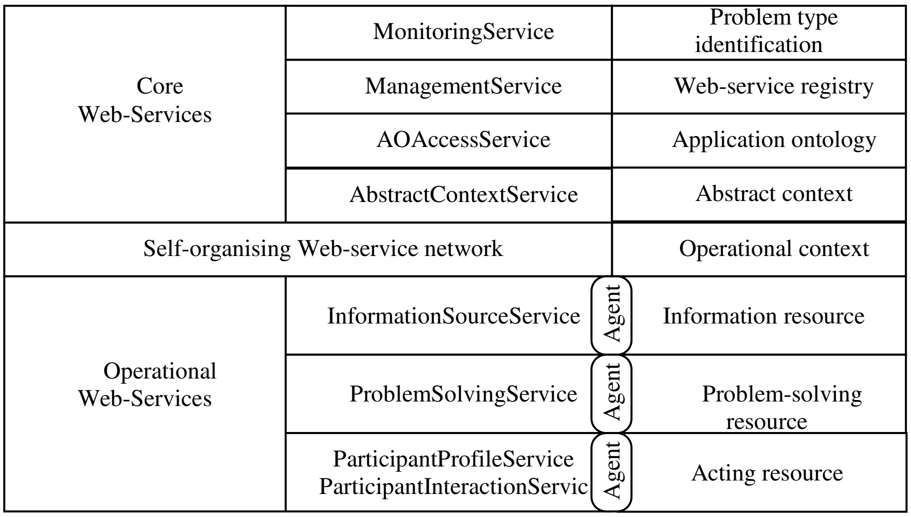


Fig. 4. Service-oriented Architecture

InformationSourceService – a set of Web-services responsible for interactions with information sources of different types and for processing information provided by these sources. The following main types of information sources are distinguished: *sensors*, *databases*, *Web-sites*, and *humans*;

ProblemSolvingService – a set of Web-services responsible for problem solving.

ParticipantProfileService creates, modifies, and updates profiles of the acting resources or, in other words, the participants of the logistic network; provides access to these profiles; collects information about the participants; in a context-based way accumulates information about the participant activities; reveals preferences of the participants;

ParticipantInteractionService – a set of Web-services responsible for support of and interactions with the participants. They communicate between the system and the participants 1) providing system messages, context-sensitive help, pictures of the current situation, results of problem solving to the participants, and 2) delivering information from the participants to the system.

5 Summary

The paper introduced an approach for self-contextualization of services for product design networks based on enterprise knowledge modelling techniques for the representation of computable context models and a technological framework using multi-agent technology. Furthermore, the requirements from product design networks to the supporting service infrastructure were identified and discussed.

The main idea behind the approach is to self-organize resources of the intelligent / information environment for the purpose to their joint actions in the design tasks. The system is built upon service-oriented architecture. Some of Web-services are used to model the resource functionalities. These Web-services self-organize a collaborative network instead of the resources. It is shown that alignment of Web-service descriptions and the application ontology at the pre-starting procedure allows the Web-services to exchange information about their needs and possibilities in terms on the ontology. The ontology-based situation model embedding the specification of problems to be solved in this situation provides the Web-services with awareness about the problems to be solved and information needed for this. As a result, the Web-services become capable to self-organize for a common purpose.

Future work will have to be of experimental and conceptual nature. From an experimental perspective, the proposed approach has to be implemented and evaluated in controlled environments or real-world cases. This will most likely lead to changes, refinements and improvements of the combined approach. The conceptual work includes to further elaborate the tight interconnection of enterprise knowledge models and self-contextualization based on the information captured in these models. In order to allow for optimal results, the knowledge to be captured and the way it is used in self-contextualization of services should be like mutually reflective views.

Acknowledgement

The paper is due to the research carried out as a part of the project funded by grant # 08-07-00264 of the Russian Foundation for Basic Research, and project # 213 of the research program "Intelligent information technologies, mathematical modelling, system analysis and automation" of the Russian Academy of Sciences. Furthermore, the work in this paper was supported by the EU-FP6-project MAPPER and the CORELIB project supported by Swedish Institute.

References

1. Billington, C., Amaral, J.: Investing in Product Design to Maximize profitability Through Postponement (2000), `http://billington.ascet.com`
2. Smirnov, A., Kashevnik, A., Levashova, T., Pashkin, M., Shilov, N.: Situation Modeling in Decision Support Systems. In: Proceedings of 2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS 2007): Modeling, EVOLUTION and Engineering, April 30 – May 3, 2007, pp. 34–39. IEEE, Waltham (2007); Electronic resource, Catalog Number 07EX1667
3. Pawlak, A., Sandkuhl, K., Cholewa, W., Indrusiak, L. (eds.): Coordination of Collaborative Engineering. 5. Intl. Workshop on Collaborative Engineering, Krakow (Poland), April 2007. Lecture Notes on Informatics, vol. P-120 (2007) ISBN 978-3-88579-214-7
4. Johnsen, S., Schümmer, T., Haake, J., Pawlak, A., Jørgensen, H., Sandkuhl, K., Stirna, J., Tellioglu, H., Jaccuci, G.: Model-based Adaptive Product and Process Engineering. In: Rabe, M., Mihók, P. (eds.) New Technologies for the Intelligent Design and Operation of Manufacturing Networks. Fraunhofer IRB Verlag, Stuttgart (2007)
5. Carstensen, A., Holmberg, L., Sandkuhl, K., Stirna, J.: Integrated Requirements and Solution Modeling: An Approach based on Enterprise Models. In: Halpin, T., Krogstie, J., Proper, E. (eds.) Innovations in Information Systems Modeling: Methods and Best Practices, IGI Publishing (2008)
6. Krogstie, J., Lillehagen, F., Karlsen, D., Ohren, O., Strømseng, K., Thue Lie, F.: Extended Enterprise Methodology. Deliverable 2 in the EXTERNAL project (2000), `http://research.dnv.com/external/deliverables.html`
7. Lillehagen, F.: The Foundations of AKM Technology. In: Proceedings 10th International Conference on Concurrent Engineering (CE) Conference, Madeira, Portugal (2003)
8. Sandkuhl, K., Smirnov, A., Shilov, N.: Configuration of Automotive Collaborative Engineering and Flexible Supply Networks. In: Cunningham, P., Cunningham, M. (eds.) Expanding the Knowledge Economy – Issues, Applications, Case Studies, IOS Press, Amsterdam, ISBN 978-1-58603-801-4
9. OASIS. Reference architecture for service oriented architecture version 1.0, 23 April (2008), `http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf` (accessed on 19.05.2009)
10. Gómez Perez, A., Benjamins, V.: Overview of knowledge sharing and reuse of components: Ontologies and Problem Solving Methods. In: Proceedings of Workshop on Ontologies and Problem Solving Methods of IJCAI (1999)
11. Grubber, T.R.: Towards principles for the design of Ontologies used for Knowledge Sharing. International Journal of Human Computer Studies (1995)
12. Motta, E.: Reusable Components for Knowledge Modelling: case studies in parametric design problem solving. IOS Press, Amsterdam (1999)

Collaboration of Metaheuristic Algorithms through a Multi-Agent System

Richard Malek

Department of Cybernetics, Faculty of Electrical Engineering
Czech Technical University in Prague, Czech Republic
malekr1@fel.cvut.cz

Abstract. This paper introduces a framework based on multi-agent system for solving problems of combinatorial optimization. The framework allows running various metaheuristic algorithms simultaneously. By the collaboration of various metaheuristics, we can achieve better results in more classes of problems.

Keywords: combinatorial optimization, metaheuristic algorithm, hybrid approach, hyper-heuristic, multi-agent system.

1 Introduction

Problems of combinatorial optimization are commonly solved by metaheuristic algorithms.

Combinatorial optimization is the process of finding one or more best (optimal) solutions in a well defined discrete problem space. Each possible solution has an associated cost. The goal is to find the solution (a configuration, a combination of values) with the lowest cost.

Best known problems are: Travelling Salesman Problem (TSP), Boolean Satisfiability Problem (SAT) or Job Shop Scheduling Problem (JSSP) [1]. All these problems are NP-complete, thus no effective algorithm exists for them.

Metaheuristic is a high-level strategy for solving optimization problem that guides other heuristics in a search for feasible solutions. Best known metaheuristics are: Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Tabu Search (TS) and many others [2] [3].

According to the No Free Lunch Theorem the average solution quality provided by any metaheuristic algorithm running on the set of all combinatorial optimization problems is statistically identical [4]. In other words, for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class.

Simple implication can be done. To be able to achieve better results in more classes of problems, we should dispose of more algorithms. Recently, this concept is denoted as a hyper-heuristic approach. An objective is that hyper-heuristics will lead to more general systems that are able to handle a wide range of problem

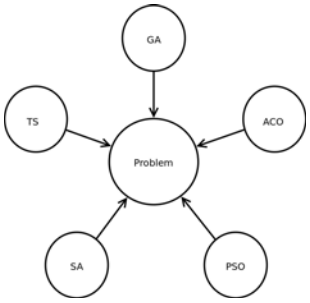


Fig. 1. The solving the combinatorial optimization problem by multiple algorithms

domains rather than current metaheuristic technology [5]. We also believe that the collaborative work of metaheuristic algorithms leads to better results than they could achieve if they worked independently. The solving of the combinatorial optimization problem by various algorithms is outlined in Figure 1.

Roughly speaking, a candidate solution is modified by more then one metaheuristic during the search process.

Intuitively, it will ultimately perform at least as well as one algorithm alone, more often perform better, each algorithm providing information to the others to help them. [6]

2 Our Concept of Collaboration

As mentioned above we are interested in using more than one metaheuristic algorithm for solving combinatorial optimization problems.

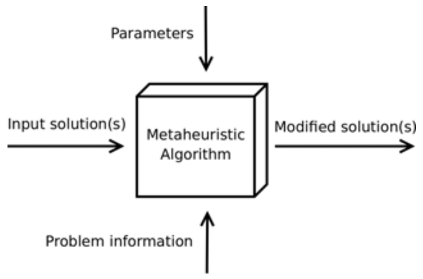
2.1 Different Metaheuristic, Different Point of View

A different metaheuristic has a different inner representation of the problem it solves and objective function (a function assigning the quality to a candidate solution) as well. Thus, different metaheuristics differ in models of the solution space they seek through and have their own point of view on the problem. Watching the problem from different points of view could be, as we believe, the main advantage of the hyper-heuristic approach.

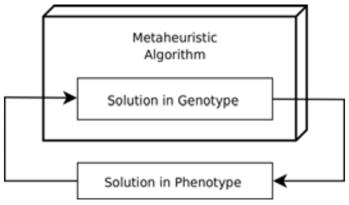
The use of different algorithms with their inherently different interfaces requires some kind of abstraction.

2.2 Abstraction of Metaheuristic Algorithm

In this section we will describe the interface all metaheuristics can be handled through. This interface unifies an access to all metaheuristics. We apprehend the metaheuristic algorithm as a black box defined by its inputs and outputs. Each algorithm has to be initiated and its parameters have to be set as well. Then it



(a) Algorithm abstraction model (algorithm adapter interface).



(b) Algorithm converts solution from/into phenotype and genotype representation.

Fig. 2.

takes candidate solutions on input, modifies them in a particular way, and puts the solutions on output. See Figure 2(a).

The inner representation of the candidate solution is the first thing we have to deal with during the algorithm unifying process. Such a representation is algorithm-specific. Dealing with various metaheuristics, their different solution representations have to be mutually transformed. However, each problem can be described by algorithm-independent information. When we use a term from genetics, a solution of a given problem can be described by its phenotype [7][8]. It is a general description of the solution–the form that can be transformed into any other algorithm-specific representations.

Thus, according to our abstraction, an algorithm takes the candidate solutions in phenotype representation on its input. It converts the solution into its inner representation–genotype. After processing, the solution is converted back to the phenotype representation and put on the output. See Figure 2(b). This evolves chaining of any algorithms that can do the phenotype/genotype mapping.

From the implementation point of view, the metaheuristic which is added is kept unchanged. Instead of modifying different metaheuristic interfaces into our one (outlined in Figure 2(a)), the only thing one has to do is to implement an adapter for each new metaheuristic. Generally, the adapter is something what converts one interface into another. Then, each metaheuristic is hidden behind the unified adapter interface (see Figure 3).

2.3 Population of Candidate Solutions

In our concept, the results of work of particular metaheuristics are shared through a global population of candidate solutions. We denote the place, where the population is located, as a solution pool. The candidate solutions are loaded from and stored in the solution pool. Each metaheuristic takes one or more solutions from and after it finishes, puts them back into solution pool. Simply, candidate solutions are improved by many different metaheuristics and population of candidate solutions is being evolved.

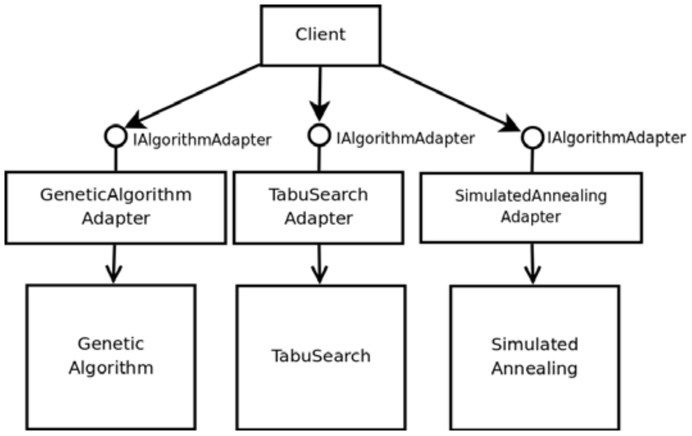


Fig. 3. Handling different metaheuristics (Genetic Algorithm, Tabu Search, Simulated Annealing) through the adapters

3 System Architecture

3.1 The Use of Multi-Agent System

Recently, multi-agent systems (MAS) have been used for solving a wide scale of artificial intelligence problems. For our purposes, the use of multi-agent system is very feasible as well. By using A-Globe [9] multi-agent system framework, we gained a lot of functionality for free (agent skeletons, message passing, agent addressing, conversation protocols, etc.).

According to our concept (presented in previous section), the system can be immediately decomposed into blocks such as particular metaheuristics, (global) solution pool and others. Each block is handled by its agent.

Our concept of collaboration of metaheuristic agents is similar to MAGMA [10] approach conceived as a conceptual and practical framework for metaheuristic algorithms.

3.2 Agents

There are following agent types in our system: a problem agent, a solution pool agent, an algorithm agent and an adviser agent.

Figure 4 shows a system configuration with six agents. Arrows represent the agent communication (will be describe in next section).

Problem Agent - the agent is the entry point of our system. It can receive a request for the optimization task either from the user or from an agent of other system. The current implementation of the agent reads problem data from a configuration file. The agent initializes all other agents by sending the *init* message. It can receive two types of messages:

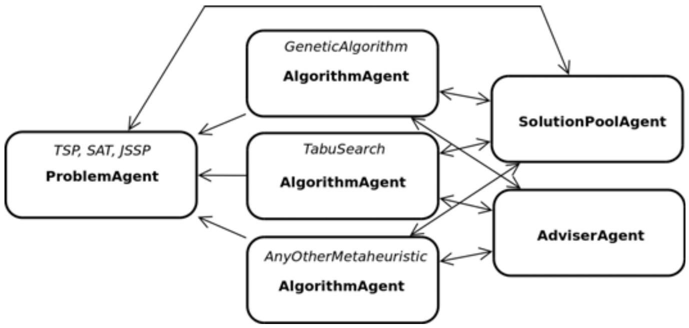


Fig. 4. Agents solving a problem

– *request* messages for initial solutions (the agent can generate initial solution since it has information about the problem)
– *inform* message when new best solution is found by the algorithm agent.

The agent also measures the overall optimization time and after timeout it sends *done* message to all agents and they finalize their work.

SolutionPool Agent - the agent manages the population of candidate solutions and provides solutions to all algorithm agents. First, it asks problem agent for an initial population, then it receives messages from algorithm agents with requests either for loading or storing solutions. It can initialize itself after receiving *init* message as well.

Algorithm Agent - the agent which disposes of a particular metaheuristic algorithm. It is responsible for:

– obtaining the algorithm parameter settings from the adviser agent
– asking the solution pool agent for candidate solution(s)
– running the metaheuristic algorithm with received parameters and solution(s)
– sending the best solution found to the problem agent after the metaheuristic algorithm is finished
– sending the solutions back to the solution pool agent
– sending a report on previous algorithm run to the adviser agent

These actions are performed until *done* message is received from the problem agent.

Adviser Agent - the agent provides parameter settings for the metaheuristic algorithms and receives reports on algorithm runs. This agent is an object of interest for future work (reasoning).

3.3 System Flow and Agent Messaging

In previous section we described agents and shortly messages they send and receive. Now, we will describe a typical system flow and messages again. Figure 5

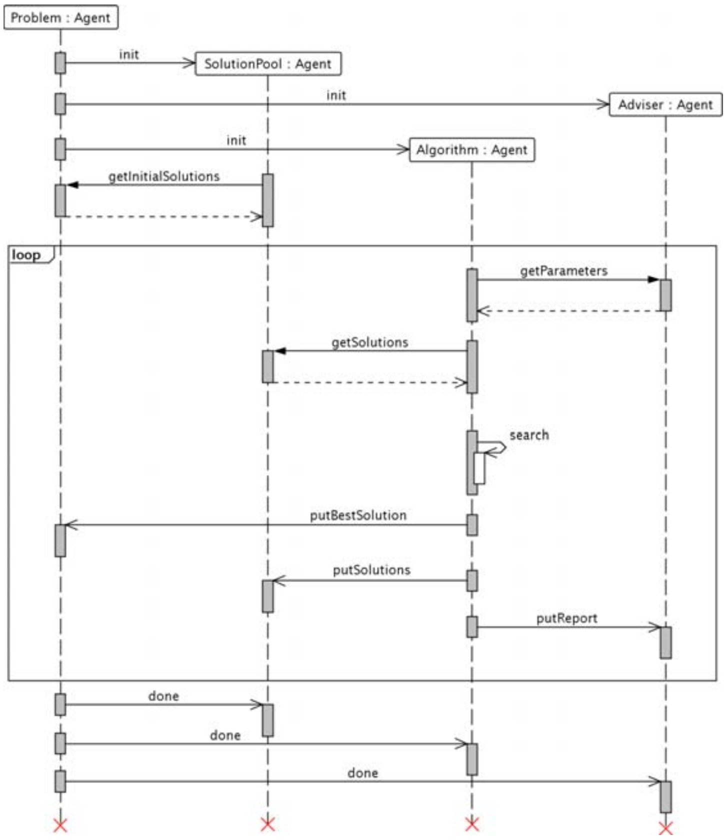


Fig. 5. Sequence diagram of agent message passing

illustrates message passing among agents as time flows. There is only one algorithm agent in the figure but the number is not limited.

1. The problem agent reads a configuration file with the information about agents it has to create. Default parameter settings for algorithm agents are included in the configuration file as well.
2. The problem agent creates all other agents and prepares (by generating or reading from file) an initial global population.
3. The solution pool agent queries the problem agent and it sends it the initial population back.
4. Once the algorithm agent is created (anytime), it asks the adviser agent for metaheuristic parameters.
5. According to the obtained parameters, the algorithm agent asks the solution pool agent for solutions to work with.
6. The algorithm agent runs its metaheuristic algorithm.
7. After finishing, the algorithm agent sends the best solution to the problem agent and other solutions back to the solution pool agent. The algorithm agent also sends a search report to the adviser agent.

8. The algorithm agent continues by the step 4.
9. After timeout the problem agent sends *done* message to all agents.

3.4 System Overview

The overall system architecture can be seen from two points of view.

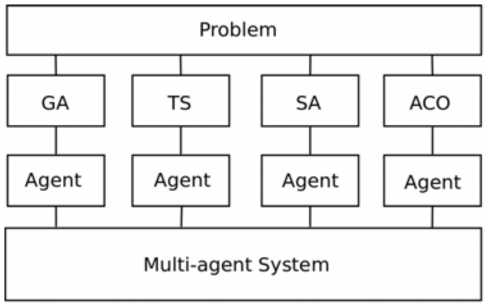
Component layer point of view (see Figure 6(a)).

1. **Problem** - the package containing all problem-dependent implementations.
2. **Metaheuristics** - the package containing all metaheuristics in their original implementations.
3. **Agents** - the package with agents running within the multi-agent system.
4. **Multi-agent system** A-Globe [9] multi-agent system.

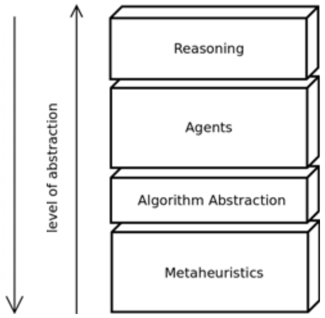
Level of abstraction point of view (see Figure 6(b)).

1. **Reasoning Layer** - the package containing a parameter reasoning implementation (see section 5)
2. **Agent Layer** - the package contains implementation of all agents mentioned above and their communication model as well.
3. **Algorithm Abstraction Layer** - the package Layer contains implementation of adapters of all metaheuristics we have at disposal.
4. **Metaheuristic Layer** - the package containing all metaheuristics in their original implementations.

From the problem perspective, each layer represents a different level of abstraction. The higher level means more general tasks.



(a) Component layer point of view



(b) Level of abstraction point of view

Fig. 6. System overview

3.5 Main Advantages of the Approach Based on Multi-Agent System

Extensibility System architecture based on multi-agent system paradigm simplify extensibility of the whole system. No inner complex bindings among components, just agents and the communication model. Adding another system feature is just about the implementation of a new agent.

Scalability is a feature of MAS allowing us freely to add or remove agents even during runtime. There are no limits (except HW limitations) for the number of algorithm agents we can run. Practically, one algorithm per one processor seems to be reasonable. Otherwise distribution of MAS comes into account.

Distributability Multi-agent systems typically contain agent containers–platforms which can be distributed on remote machines and utilize host resources. Communication among agents from different platforms is the same as within one platform. This allows the creation of large logical multi-agent system running on many computers.

4 Experiments

For experiment purposes, we have implemented Genetic Algorithm (GA) and Tabu Search (TS) metaheuristics and tested them on Traveling Salesman Problem (TSP). We run GA and TS on several TSP benchmark instances with parameters experimentally set in previous experiments (not mentioned here). Each algorithm was run 10x and the best and the average values of found solutions have been recorded. Then we run TS and GA simultaneously. In all three cases, the algorithms were run for the same time–10 minutes. The results are presented in Figure 7.

We have got the best results by algorithm collaboration in most cases and we found the results promising for future work.

		GA		TS		GA+TS	
instance	optimal	min	avg	min	avg	min	avg
eil51	426	478	512	430	438	**428**	**435**
berlin52	7542	8261	9329	8007	8139	**7544**	**7693**
ch130	6110	11300	12074	9315	9644	**7303**	**7829**
ch150	6528	13564	14837	9754	10224	**8118**	**8654**
d198	15780	38177	44957	25393	28771	**24235**	**26074**
gil262	2378	9374	9670	4434	4765	4 **223**	**4462**
lin318	42029	216098	228465	99373	103442	**90673**	**100091**
rd400	15281	92349	97823	**39258**	44567	40682	**44239**
pr439	107217	791144	846171	**345676**	418296	386018	**413126**
u574	36905	345138	352248	**156576**	**207538**	214900	216475

Fig. 7. Table of results. Best values are printed bold

5 Future Work

5.1 Reasoning

Adviser Agent. The effectiveness of a metaheuristic algorithm solving a given problem depends on parameters that were set. Values of the parameters are commonly determined experimentally and they are not changed during the algorithm runtime. The problem is that "optimal" parameters may differ with different data sets. Instead of determining the parameters by hands, it would be nice to set the parameter values automatically. In other words, we are interested in the evolution of parameters. That's what the adviser agent should do (see section 3.2 for information about the role of the adviser agent).

We remark that algorithm agents ask the adviser agents for metaheuristic parameters and send back reports describing metaheuristic algorithm runs. According to the reports, the adviser answers for the parameter requests. It may answer randomly or it can create its inner model supporting the parameter reasoning.

During the algorithm runtime a lot of information may be tracked. For example: the number of solution improvements, the number of iterations without change, the change of average solution quality, etc.All the information may indicate if the algorithm parameters were set properly. The idea is such that we will track all the runtime information and when the algorithm stops, the information will be processed and new parameter values will be set accordingly.

Metaheuristic Competition. Because of limited resources and possibly unlimited number of algorithm agents, we have to select which combination of metaheuristics and in how many instances (algorithm agents) is employed for solving a given optimization problem. In this context a competition of algorithm agents is intended.

6 Conclusion

In this paper we have presented our concept and the implementation solving the problems of combinatorial optimization by employing various metaheuristic algorithms. Being aware of No Free Lunch theorem, our concept is based on combination of particular metaheuristics to achieve better results for wide scale of problems and problem instances as well.

An approach of collaboration of different metaheuristic algorithms through a multi-agent system has been introduced. To be such a implementation possible to realize, an abstraction of the metaheuristic algorithm had to be done.

The metaheuristic algorithm is apprehended as a black box defined by its inputs and outputs. Each algorithm has to be initiated and its parameters have to be set as well. Then it takes candidate solutions on input, modifies them in a particular way, and puts the solutions on output. On input and output it handles with candidate solutions in phenotype (algorithm independent) representation.

This abstraction allows algorithm chaining when the output of one algorithm can be put on the input of the other one.

For this purpose, an architecture based on multi-agent system paradigm has been chosen. Algorithm agents and the solution pool agent are the base of the system. Main advantages of the multi-agent system are: scalability, extensibility and distibutability.

Two algorithms have been implemented (GA, TS) and tested on Travelling Salesman Problem. Experiments have shown that improvements of solution quality obtained by the metaheuristic collaboration are notable. However, other detailed experiments have to be done.

Automated parameter setting of particular algorithms was discussed and the adviser agent solving the parameter adaptation was proposed. It is also the object of interest for future work.

References

1. Skiena, S.S.: The Algorithm Design Manual. Springer, Heidelberg (1998)
2. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, Frome (2008)
3. Yagiura, M., Ibaraki, T.: On metaheuristic algorithms for combinatorial optimization problems. Transactions of the Institute of Electronics, Information and Communication Engineers (J83-D-1) 3–25
4. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1, 67–82 (1997)
5. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology (2003)
6. Talbi, E.-G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(5), 541–564 (2002)
7. Fogel, D.B.: Phenotypes, genotypes, and operators in evolutionary computation, pp. 193–198 (1995)
8. Back, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol (1997)
9. Sislak, D., Rehak, M., Pechoucek, M.: A-globe: Multi-agent platform with advanced simulation and visualization support. Web Intelligence. In: IEEE/WIC/ACM International Conference, pp. 805–806 (2005)
10. Roli, A., Milano, M.: Magma: A multiagent architecture for metaheuristics. IEEE Trans. on Systems, Man and Cybernetics - Part B 34, 2004 (2002)

Functional Integrity of Multi-agent Computational System Supported by Component-Based Implementation

Kamil Piętak, Adam Woś, Aleksander Byrski, and Marek Kisiel-Dorohinicki

AGH University of Science and Technology, Kraków, Poland
{kpietak,awos,olekb,doroh}@agh.edu.pl

Abstract. In the paper a formalism is proposed to describe the hierarchy of multi-agent systems, particularly suitable for the design of a certain class of distributed computational intelligence systems. The notions of algorithms and dependencies among them are introduced, which allow for the formulation of functional integrity conditions for the whole system. General considerations are illustrated by modeling a specific case of an evolutionary multi-agent system. Component techniques introduced in *AgE* computing environment facilitate the implementation of the system in such a way that algorithm dependencies are represented as contracts, which support checking of the system's functional integrity.

Keywords: functional integrity, components, mutli-agent systems.

1 Introduction

The notion of *functional integrity* may be understood as the ability to fulfill functional requirements in a complex system. In agent-based environments, it may be really difficult to check whether the set of cooperating but autonomous agents is able to achieve the global goal of the system (solve the problem) [1]. Due to their dynamic nature, it may require to forecast and manage different critical situations, such as sudden breakdown of hardware [2]. But first of all agents must be able to cooperate with one another, which requires adequate infrastructure addressing interoperability issues [3].

The paper focuses on a specific class of agent systems, which use computational intelligence paradigms – particularly hybrid techniques based on the concept of decentralized evolutionary computation [4]. These systems consist of a hierarchy of agents and nested multi-agent subsystems, which should use compatible structures and mechanisms to be able to work together. What is more, some agents perform similar tasks, but work with different structures and mechanisms. Thus from the software engineering perspective it may be said that the system is decomposed into particular agents, but a single agent implementation is too complex to serve as an assembly unit. In fact agents implementations may be further decomposed into functional parts (components), which are replaceable, as long as they are compatible to one another, even when used by different agents (this ensures agents interoperability at implementation level).

As it was discussed in [3], both agents and components can be considered in software development as assembly units, which implement in various ways the concept of responsibility delegation. However, agents-based technology focuses more on executing complex tasks in a community to achieve defined goals and, on the other side, component-based technology is rather aimed at reusability and integrity aspects of software development [5]. Indeed, it seems that component-oriented approach can be also successfully exploited in agent-based systems for assembling parts of agents implementation and checking the integrity of the system.

To facilitate the design of the systems under consideration a dedicated formalism was proposed in [6]. The goal of this paper is to show how it may be extended to formulate the *functional integrity* conditions of the system, and how these conditions may be supported by component technology. In the course of paper the notion of algorithms (used by agents for performing actions) is introduced, as well as additional relation (dependency) imposing the existence of algorithms required by actions, as well as other algorithms. Agents and algorithms are implemented and provided to *AgE* computing environment[1] as components. Relations between them are implemented as component contracts handled by the configuration engine of *AgE* system, which is also responsible for the verification of the system functional integrity.

The paper begins with the presentation of a formalism, which allows for the formulation of functional integrity conditions of the system. The case of an evolutionary multi-agent system is discussed as an illustration in the next section. Finally the realization of these concepts using component techniques in *AgE* computing environment is shown and some conclusions are drawn.

2 Functional Integrity of a Computing MAS

The model proposed in [6] defines an agent as a tuple:

$$AG \ni ag = \langle id, tp, dat_1, \ldots, dat_n \rangle \tag{1}$$

where $id \in ID$ is a unique identifier of an agent[2], $tp \in TP$ denotes the type of an agent (depending on its type, an agent is equipped with specific data and may perform specific actions), and $dat_i \in DAT_i, i = 1, \ldots, n$ represents problem-dependent data (knowledge) gathered by an agent.

According to [6] a multi-agent system includes agents, actions to be executed by the agents, and the environment represented by some common data, which may acquired by the agents. This definition must be extended here with a set available algorithms:

$$AS \ni as = \langle Ag, Act, Alg, qr_1, \ldots, qr_m \rangle \tag{2}$$

[1] http://age.iisg.agh.edu.pl/

[2] For each element of the model its domain, which is a finite set of possible values, is denoted by the same symbolic name in upper case, e.g. *ID* is the set of all possible agent identifiers.

where $Ag \subset AG$ is the set of agents of as, $Act \subset ACT$ describes actions that may be performed by the agents of as, $Alg \subset ALG$ is the set of algorithms available in as, and $qr_i \in QR_i, i = 1, \ldots, m$ denote queries providing data (knowledge) available for all agents in as.

An agent may provide an environment for a group of other agents, which by themselves constitute a multi-agent system, which is essentially different then the one of the "parent" agent. These nested (multi-agent) subsystems introduce a tree-like structure, which will be further referred as *physical hierarchy* of agents. The relation $\gamma : AS \rightarrow AG \cup \{\varnothing\}$ identifies an agent that provides the environment for a particular agent system. For details see [6].

In the space of types, a subsumption relation "$\preceq$" $\subset TP \times TP$ is defined, introducing a partial order in TP. In terms of this relation, $A \preceq B : A, B \in TP$ means that A is a subtype of B.

Agents may perform actions in order to change the state of the system. An action is defined as the following tuple (Hoare's triple equivalent [7]):

$$ACT \ni act = \langle tp, pre, post \rangle \tag{3}$$

where $tp \in TP$ denotes the type of agents allowed to execute the action (only agents of the type tp and descendant types – according to the "$\preceq$" relation – may perform the action); $pre \in X$ is the state of the system which allows for performing action act; $post \in X \times X$ is the relation between the state of the system before and after performing action act.

Actions may depend on algorithms, i.e. in order to perform an action, one or more algorithms may be needed. This dependency is described by the following relation:

$$\text{"}\rightsquigarrow\text{"} \subset ACT \times ALG \tag{4}$$

For algorithms there is also a subsumption relation "$\preceq$" defined, which states whether one algorithm is a specialization of another:

$$\text{"}\preceq\text{"} \subset ALG \times ALG \tag{5}$$

Relation "$\preceq$" introduces a partial order in ALG (it is reflexive, transitive and antisymmetric), i.e. if $A1 \preceq A$ then $A1$ can be used in place of A when needed.

When an action is about to be performed, a subset of algorithms is selected from Alg according to the $\preceq$ relation and any further restrictions described below. These algorithms are said to be "available" in the environment for the execution of a particular action[3]. For example, if action act depends on algorithm A, and in a particular system $\exists!\, subA \in Alg : subA \preceq A$, then when the action is executed, it uses algorithm $subA$.

Further dependencies between algorithms used in the system may be described using the following relation:

$$\text{"}\rightsquigarrow\text{"} \subset ALG \times ALG \tag{6}$$

[3] A discussion on how this selection is realized in AgE is provided in section 4.

If algorithm A depends on algorithm B ($A \rightsquigarrow B$) it means that B or its subtype is needed for A to function properly, and must be available in the system. Relation "$\rightsquigarrow$" allows for defining families of algorithms that are designed to be used together (i.e. if one of them is selected by the environment for the execution of a particular action, then other algorithms from the same family are also selected).

For example, let us assume that the set of all known algorithms is $ALG = \{A, A1, B, B1, B2, C\}$, where $A1 \preceq A$, $B1 \preceq B$ and $B2 \preceq B$, the set of all known actions is $ACT = \{act\}$ and that $A1 \rightsquigarrow B1$ and $A1 \rightsquigarrow C$. Moreover, let us consider an AS with the set of available algorithms $Alg = \{A1, B1, B2, C\}$. When action act dependent on both A and B ($act \rightsquigarrow A$, $act \rightsquigarrow B$) is to be performed, the set of algorithms that must be available for its execution is determined. For this set, $A1$ is selected as the only algorithm subsuming A, and $B1$ is selected because $A1$ depends on it, even though $B2$ could be selected as well if dependencies between algorithms were not considered. Moreover, C is selected because $A1$ depends on it, even though act does not depend on C explicitly.

The proposed formalism allows to formulate the conditions of *functional integrity* of the whole system. The system is functionally integral when the following coherency conditions are true for all agent subsystems $AS \ni as = \langle Ag, Act, Alg, qr_1, \ldots, qr_m \rangle$:

$$\forall\, act \in Act\, [\, (\exists\, alg \in ALG : act \rightsquigarrow alg) \Rightarrow (\exists\, alg_c \in Alg : alg_c \preceq alg)\,] \tag{7}$$

$$\forall\, alg_1 \in Alg\, [\, (\exists\, alg_2 \in ALG : alg_1 \rightsquigarrow alg_2) \Rightarrow (\exists\, alg_g \in Alg : alg_g \preceq alg_2)\,] \tag{8}$$

i.e. for each action that may be performed in the agent system ($act \in Act$), if this action depends on algorithm alg, then an algorithm alg_c subsuming alg must be available in the system (i.e. must be present in the Alg set of the system). As was mentioned before, the subsumption relation "$\preceq$" is a partial order, and therefore alg_c can equal alg because $alg \preceq alg$. Similarly, for each algorithm alg_1 that is available in the AS, if alg_1 depends on alg_2, then an algorithm subsuming alg_2 (in particular, alg_2 itself) must be also available in the agent system ($\exists\, alg_g \in Alg$).

3 Functional Integrity of an Evolutionary Multi-agent System

The idea of an evolutionary multi-agent system (EMAS) was proposed as a particular technique of decentralized evolutionary computation [8,4]. The system consists of individual agents decomposed into several subpopulations (demes). Agents possess (possibly partial) solutions of the given optimization problem. They also possess a non-renewable resource called *life energy*, which is the base of a distributed selection process. Agents exchange their energy based on the quality of their solutions (fitness). Those which gather more energy have greater chances of reproducing, and those with low energy have greater chances of dying. This energy-based selection is used instead of classical global selection mechanisms, because of the assumed autonomy of agents. Agents may also migrate to another subpopulation if they have enough energy.

The model of EMAS which follows the concepts of [6] defines two types of agents:

$$TP = \{ind, isl\} \tag{9}$$

where ind denotes the type of an individual agent (as described above), and isl — of an aggregate agent, which is introduced to manage subpopulations of individual agents (an evolutionary island).

Consequently, at the top of the physical structure of EMAS there is a system of evolutionary islands:

$$as = \langle Ag, \varnothing \rangle \quad \gamma(as) = \varnothing \tag{10}$$

where:

$$Ag \ni ag = \langle id, isl, Nb \rangle \tag{11}$$

and $Nb \subset Ag$ is the set of evolutionary islands, which is used to define the topology of migration.

Every evolutionary island ag provides an environment for the population of individual agents:

$$\forall\ ag \in Ag\ \ \exists\ as^* = \langle Ag^*, Act^*, Alg^*, findAg, findLoc \rangle : ag = \gamma(as^*) \tag{12}$$

and an individual agent is defined as:

$$Ag^* \ni ag^* = \langle id, ind, sol, en \rangle \tag{13}$$

where:

$sol \in SOL$ is the solution of the problem (usually for optimization problems $SOL \subset \mathbb{R}^n, n \in \mathbb{N}$),
$en \in \mathbb{R}^+$ is the amount of energy gathered by the individual agent,
$ACT \supseteq Act^* = \{init, migr, get, repr, die\}$ is the set of actions available in as^*:
  $init$ – initialization of agent's solution,
  $migr$ – migration of an agent from one to another subpopulation,
  get – transfer of a portion of energy from one to another agent,
  $repr$ – creation of a new agent by two parents,
  die – removing of an agent from the system,
$ALG \supseteq Alg^* = \{rand, prep, eval, recomb, mut\}$ is the set of algorithms available in as^*,
$findAg : 2^{AG} \to \mathcal{M}(AG)$ is the query which allows to choose the neighboring individual agent (another agent present in the same system),
$findLoc : 2^{AG} \to \mathcal{M}(AG)$ is the query which allows to choose the neighboring island (using $Nb \subset Ag$).

An action of solution initialization $init$ performed by $ag^* = \langle id, ind, sol, en \rangle \in Ag^*$ is defined in the following way:

$$Act^* \ni init = \langle ind, [\tau(sol) = 0], [\tau(sol) = 1] \rangle \tag{14}$$

where $\tau : SOL \rightarrow \{0, 1\}$ is a problem-dependent function, which indicates if a solution is initialized (by returning 1) or not (by returning 0). During this process a problem-dependent algorithm $prep \in Alg^*$ is used, and therefore: $init \rightsquigarrow prep$. Also, algorithm $prep$ depends on another algorithm $rand$, necessary for generating random solutions. This is denoted as: $prep \rightsquigarrow rand$.

Actions $migr$, get, $repr$ and die were defined in [6] and will not be discussed in detail here. However, several interesting observations follow. The action of energy transfer get is performed by an individual agent depending on the quality of its solution, which is given by problem-dependent algorithm $eval$. This relation may be defined as: $get \rightsquigarrow eval$. Similarly, the action of reproduction $repr$ depends on the algorithms realizing recombination and mutation operators — $recomb$ and mut respectively. This is denoted as: $repr \rightsquigarrow recomb$, $repr \rightsquigarrow mut$.

Application of EMAS to solving concrete optimization problems requires undertaking several decisions typical for the evolutionary approach, e.g. the choice of genotype representation and adequate variation operators. As an illustration one may consider EMAS with binary representation — it requires specializations (in terms of the subsumption relation) of algorithms $prep$, $eval$, $recomb$, mut:

$binPrep \preceq prep$ denotes an algorithm necessary to generate binary solutions,
$binEval \preceq eval$ denotes an algorithm which evaluates binary solutions,
$binRecomb \preceq recomb$ denotes an algorithm responsible for recombination of two binary solutions,
$binMut \preceq mut$ denotes an algorithm responsible for mutation of a binary solution.

All these elements must be introduced into ALG and Alg^*, and the latter will be defined as:

$$Alg^* = \{rand, binPrep, binEval, binRecomb, binMut\} \tag{15}$$

Such system definition satisfies equations (7) and (8) so it is functionally integral. According to the mechanism presented in the previous section, all algorithms are applicable without changing existing actions' definitions.

4 Component Techniques for AgE Environment

The model presented is the base for the design of the core of the computing environment *AgE*[4], which is developed as an open-source project at the Intelligent Information Systems Group of AGH-UST. A system implemented on *AgE* platform is composed of agents – the main functional entities, which represent the core logic of the computation [9]. Agents are further decomposed into functional units according to *Strategy* design pattern [10]. Algorithms, as presented in the model, are implemented in the form of strategies in *AgE*. Both agents and strategies can have properties (described in more detail in [6]), which can be either simple values or references to other entities in the system.

[4] http://age.iisg.agh.edu.pl/

Actions are implemented and executed as methods of external strategies classes or the parent agent class, which represents the agent's environment. During the execution of these methods other strategies can be used to perform different activities within the action. Therefore, the dependency (described by the relation "$\rightsquigarrow$") between actions and algorithms is represented in *AgE* as a reference property to the required algorithm, preceded by @Inject annotation.

Dependency between algorithms introduced in the model maps to dependency between strategies. In different cases this relation can be represented in two ways in *AgE*.

Let us consider dependent algorithms A and B ($A \rightsquigarrow B$) represented in AgE by Java classes `A` and `B`. In the first case strategy `A` directly uses strategy `B` and the dependency relation is realized in the same way that dependency between actions and algorithms, i.e. by using a reference property marked with @Inject annotation. In the next case, strategy `A` does not directly use `B`, but `B` is required for proper processing of `A`. For example, a binary mutation strategy needs a binary initialization strategy which generates the proper type of solutions, although the initialization is not directly used by mutation. To define this kind of dependency, class `A` must be preceded by annotation @Require(B.**class**).

An example code of a strategy with dependencies is shown below:

```
@Require(C.class) public class A {
    @Inject @PropertyField(name="b")
    private B b;
}
```

Strategy `A` directly uses strategy `B` and requires strategy `C` for proper processing. Therefore it defines two dependencies, the first by an annotated reference property (for strategy `B`) and the second by adding @Require annotation to class definition (for strategy `C`).

AgE was designed with the emphasis on achieving the main advantages of component-oriented techniques, i.e. independent development, reusability and elementary contracts [7]. Its realization is vastly supported by dependency injection pattern, an implementation of the inversion of control paradigm proposed by Robert Martin [11] and later popularized by Martin Fowler[5], and by utilizing the freely available PicoContainer framework[6].

Component-oriented techniques are exploited to implement the process of automatic assembly of different agents structures with dependant strategies. The input configuration (in XML file format) together with agents and strategies classes with described annotations are used to initialize computing environment and provide appropriate instances in runtime.

The approach allows for creation of fully-initialized components, with all dependent components injected. Moreover, late binding by a container allows for runtime injection which facilitates third-party development of the components. The annotations describing component dependencies, as presented above,

[5] http://martinfowler.com/articles/injection.html

[6] http://www.picocontainer.org

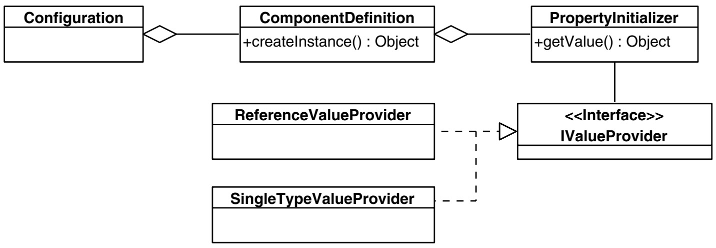


Fig. 1. Configuration model

together with class's public methods treated as component's operations, may be perceived as a requirement closely related to component contracts as proposed by Szyperski [7].

The decision which component should be provided to the other one is made during instantiating particular components based on the configuration model shown in Fig. 1. The functional integrity of the system as defined by (7) and (8) is ensured by verifying the configuration and components requirements – dependencies are processed in order to check if all required components are available in the classpath, no conflicts occur between components, and all requirements are fulfilled. Verification is also performed during each request for instantiating a component, which ensures that no inconsistent unit will be created.

The configuration model describes agents and strategies by defining a set of `ComponentDefinition` objects, aggregated in one `Configuration`. Each definition can have a list of `PropertyInitializer`s which – each assigned to an agent's or strategy's single property – are responsible for their initialization. To facilitate this initialization AgE uses `IValueProvider` objects, which are responsible for providing concrete values to properties. These values can be references (represented by `ReferenceValueProvider`s) or simple type values (represented by `SingleTypeValueProvider`s).

The process of system initialization is divided into two steps. In the first step, the configuration model is created from an XML file with a well-defined structure[7] and a process of verification is executed.

In the next step, a fully initialized system is created, containing agents with already associated strategies. The advantage of this solution is that one implementation of an agent can perform actions which use different realizations of strategies. Therefore a wide variety of possible computations can be performed without changing an agent's class, i.e. without recompiling the system. One thing remains to be done, i.e. to change the XML configuration file.

After being built from an XML configuration file, component definitions are registered in IoC containers (class `PicoContainer`) using special adapters of class `CoreComponentAdapter`, both shown in Fig. 2. A request to create a hierarchy of components (agents and strategies) is directed to an IoC container, which in turn delegates the creation of specific components to adapters assigned to component

[7] http://age.iisg.agh.edu.pl/xsd/age-2.3.xsd

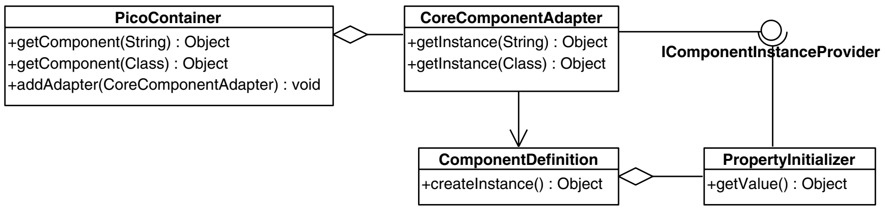

PicoContainer
+getComponent(String) : Object
+getComponent(Class) : Object
+addAdapter(CoreComponentAdapter) : void
CoreComponentAdapter
+getInstance(String) : Object
+getInstance(Class) : Object
IComponentInstanceProvider
ComponentDefinition
+createInstance() : Object
PropertyInitializer
+getValue() : Object

Fig. 2. Dependency Injection pattern in AgE

definitions. In the method `ComponentDefinition.createInstance`, the definition creates an instance of a component it describes. This method is responsible for properly initializing all the component's properties, both simple values and its dependencies on other components. In order to initialize the dependencies correctly, the definition can retrieve (via the `IComponentInstanceProvider` interface) an instance of a required component (either by name or by type) from the IoC container associated to it. The retrieval of required values and assignment to component properties is done by specialized `PropertyInitializer` objects, one for simple type values and one for references to other components.

5 Conclusions

The formalism proposed in the paper is solely used for design, and that is why such details as the precise definition of the system state space or state transition functions were not taken into consideration. The notion of algorithms and dependency relations (imposing the existence of algorithms required by actions and other algorithms) were introduced to define functional integrity conditions for the system. So far the model can be fully mapped to the concepts present in *AgE* computing environment. Also component-based approach proved to be a convenient and flexible technique supporting the assembly of a particular system according to the provided configuration, and its validation with respect to the proposed functional integrity rules. Moreover, a prototype graphical configuration editor was created based on the presented solution. It uses the proposed techniques for suggesting available assembly options according to components contracts.

Further research should allow to extend the model to cover other computation intelligence techniques based on agent paradigm, such as iEMAS (immunological evolutionary multi-agent system) [12] or HGS (hierarchical genetic search) [13]. The mapping between the proposed formalism and existing models describing these techniques will be provided. The current focus of *AgE* development is to fully implement the verification logic allowing to check the integrity rules for delivered components. Also, because the implementation of AgE framework continues, the formalism will surely be updated in the near future.

Acknowledgments. The authors would like to express their gratitude to Mr. Tomasz Kmiecik, a former student of AGH-UST, for his invaluable contribution to the design and implementation of dependency injection mechanisms in AgE.

References

1. Cetnarowicz, K., Dobrowolski, G., Kisiel-Dorohinicki, M., Nawarecki, E.: Functional integrity of mas through the dynamics of the agents' population. In: Proc. of 3nd Int. Conf. on Multi-Agent Systems (ICMAS 1998). IEEE Computer Society Press, Los Alamitos (1998)
2. Jamont, J.P., Ocello, M.: Using self-organization for functional integrity maintenance of wireless sensor networks. In: Proc. of the International Conference on Intelligent Agent Technology IAT 2003, Washington, DC, USA. IEEE, Los Alamitos (2003)
3. Bergenti, F., Gleizes, M.P., Zambonelli, F.: Methodologies and Software Engineering for Agent Systems. Kluwer Academic Publishers, Dordrecht (2004)
4. Kisiel-Dorohinicki, M.: Agent-oriented model of simulated evolution. In: Grosky, W.I., Plášil, F. (eds.) SOFSEM 2002. LNCS, vol. 2540, pp. 253–261. Springer, Heidelberg (2002)
5. Krutisch, R., Meier, P., Wirsing, M.: The agentComponent approach, combining agents, and components. In: Schillo, M., Klusch, M., Müller, J., Tianfield, H. (eds.) MATES 2003. LNCS (LNAI), vol. 2831, pp. 1–12. Springer, Heidelberg (2003)
6. Byrski, A., Kisiel-Dorohinicki, M.: Agent-based model and computing environment facilitating the development of distributed computational intelligence systems. In: Proc. of the Computational Science - ICCS 2009, 9th International Conference, Baton Rouge, U.S.A., May 25 - 27, 2009. Springer, Heidelberg (2009)
7. Szyperski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
8. Cetnarowicz, K., Kisiel-Dorohinicki, M., Nawarecki, E.: The application of evolution process in multi-agent world (MAW) to the prediction system. In: Proc. of 2nd Int. Conf. on Multi-Agent Systems (ICMAS 1996). AAAI Press, Menlo Park (1996)
9. Kisiel-Dorohinicki, M.: Agent-based models and platforms for parallel evolutionary algorithms. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3038, pp. 646–653. Springer, Heidelberg (2004)
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley Professional, Reading (1995)
11. Martin, R.C.: The dependency inversion principle. C++ Report 8(6), 61–66 (1996)
12. Byrski, A., Kisiel-Dorohinicki, M.: Immunological selection mechanism in agent-based evolutionary computation. In: Klopotek, M., Wierzchon, S., Trojanowski, K. (eds.) Proc. of the Intelligent Information Processing and Web Mining IIS IIPWM 2005, Gdansk, Poland. Advances in Soft Computing. Springer, Heidelberg (2005)
13. Schaefer, R., Kołodziej, J.: Genetic search reinforced by the population hierarchy. Foundations of Genetic Algorithms 7 (2003)

On the Empirical Evaluation of an Interdisciplinary Framework for Automated Negotiation

Fernando Lopes[1], A.Q. Novais[1], and Helder Coelho[2]

[1] National Research Institute (INETI), Modelling and Simulation Dept., Lisbon, Portugal
{fernando.lopes,augusto.novais}@ineti.pt
[2] University of Lisbon, Computer Science Department, Lisbon, Portugal
hcoelho@di.fc.ul.pt

Abstract. Negotiation is an important and pervasive form of social interaction. The task of designing and implementing autonomous agents with negotiation competence is usually understood as involving the consideration of insights from multiple relevant research areas to integrate different perspectives on negotiation. Hence, this paper uses both game-theoretic techniques and methods from the social sciences as a basis to develop autonomous negotiating agents. It employs game-theoretic techniques to define equilibrium strategies for the bargaining game of alternating offers, and formalizes a set of negotiation strategies and tactics studied in the social sciences and frequently used by human negotiators. This paper also lays the foundation for performing an experiment to investigate the behaviour of autonomous agents equipped with the strategies, paying particular attention to the quality and the cost of bargaining.

Keywords: Multi-agent systems, Automated negotiation, Bargaining.

1 Introduction

Negotiation is an important and pervasive form of social interaction – it may involve two parties (bilateral negotiation) or more than two parties (multilateral negotiation), and one issue (single-issue negotiation) or many issues (multi-issue negotiation). Parties are the agents (or groups of agents with common interests) that participate in negotiation and issues are the resources to be allocated or the considerations to be resolved [15].

Negotiation is usually understood as proceeding through distinct phases or stages. A phase is a coherent period of interaction characterized by a dominant constellation of communicative acts that serves a set of related functions in the movement from initiation to a resolution of a dispute. Phase models provide a narrative explanation of the negotiation process, $i.e.$, they identify sequences of events that constitute the story of negotiation. Most models fit into a general structure of three phases [7]: a beginning or initiation phase, a middle or problem-solving phase, and an ending or resolution phase. The initiation phase focuses on preparation and planning for negotiation (usually referred to as pre-negotiation) – it is marked by each party's efforts to emphasize points of difference and posture for positions. The problem-solving phase seeks a solution for a dispute – it is characterized by extensive interpersonal interaction, strategic maneuvers, and movement toward a mutually acceptable agreement. The resolution phase focuses on details and implementation of a final agreement.

Automated negotiation promises a higher level of process efficiency and a higher quality of agreements (when compared to traditional, face-to-face negotiation). The demands for systems composed of agents that are owned by different individuals or organizations and are capable of reaching agreements through negotiation are becoming increasingly important and pervasive. Examples, to mention a few, include:

1. the industrial trend toward agent-based supply chain management – agents negotiate the conditions for purchasing raw materials, decide and execute the scheduling, and negotiate the terms under which the final products are delivered;
2. the business trend toward virtual enterprises – dynamic alliances of small, agile enterprises which together can take advantage of economies of scale;
3. the pivotal role that e-commerce is increasingly assuming in organizations – e-commerce offers opportunities to improve (make faster, cheaper, and more agile) the way that businesses interact with both customers and suppliers.

Artificial intelligence (AI) researchers have paid a great deal of attention to automated negotiation over the past decade and a number of prominent models have been proposed in the literature. Some researchers have followed a theoretical or formal mathematical perspective and have mainly attempted to develop negotiation theories. To this end, they have drawn heavily on game-theoretic and economic methods (see, e.g., [3,6]). On the other hand, various researchers have followed a practical or system-building perspective and have mainly attempted to develop computer systems with negotiation competence. To this end, they have drawn heavily on social sciences techniques for understanding interaction and negotiation (see, e.g., [4,12]). Overall, most existing models primarily use either game-theoretic techniques or methods from the social sciences as a basis to develop autonomous agents with negotiation competence, and largely ignore the integration of the results from both research areas. Yet, the task of designing and implementing autonomous negotiating agents is usually understood as involving the consideration of insights from multiple relevant research areas to integrate different perspectives on negotiation (see, e.g., [1,5]).

This paper argues that an interdisciplinary approach towards the development of autonomous negotiating agents is possible and highly desirable – game-theoretic (strategic) and behavioural negotiation theories can mutually reinforce each other and lead to more comprehensive and richer models. As a starting point for this research effort, we have already proposed a model for autonomous agents that handles two-party multi-issue negotiation [8,9,10,11]. The model describes equilibrium strategies for the bargaining game of alternating offers and formalizes a set of strategies and tactics frequently used by human negotiators. On the one hand, it considers two fully informed agents about the various aspects of the bargaining game and employs game-theoretic techniques to define equilibrium strategies. On the other hand, it considers two incompletely informed agents and formalizes a set of negotiation strategies studied in the social sciences. The strategies are based on rules-of-thumb distilled from human behavioural practice and need to be empirically evaluated to determine precisely how they behave in different situations. To this end, this paper lays the foundation for performing an experiment to investigate the behaviour of agents equipped with the negotiation model, paying particular attention to the quality and the cost of bargaining.

2 The Negotiation Model

This section briefly introduces the negotiation model (see [8,9,10,11] for an in-depth discussion). Let $Ag=\{ag_1, ag_2\}$ be the set of autonomous negotiating agents. Let $Agenda=\{is_1,\ldots,is_n\}$ be the negotiating agenda – the set of issues to be deliberated. The issues are quantitative variables, defined over continuous intervals. The priority pr_{il} of an agent $ag_i \in Ag$ for each issue $is_l \in Agenda$ is a number that represents its order of preference. The weight w_{il} of is_l is a number that represents its relative importance. The limit lim_{il} or resistance point is the point where ag_i decides that it should stop the negotiation. The target point trg_{il} or level of aspiration is the point where ag_i realistically expects to achieve a settlement.

2.1 The Negotiation Protocol and Time Preferences

The negotiation protocol is an alternating offers protocol [13]. Two agents or players bargain over the division of the surplus of $n \geq 2$ issues (goods or pies). The players determine an allocation of the issues by alternately proposing offers at times in $\mathcal{T}=\{1,2,\ldots\}$. This means that one offer is made per time period $t \in \mathcal{T}$, with an agent, say ag_i, offering in odd periods $\{1,3,\ldots\}$, and the other agent ag_j offering in even periods $\{2,4,\ldots\}$. The negotiation procedure, labelled the “joint-offer procedure”, involves bargaining over the allocation of the entire endowment stream at once. An offer is a vector $(x_1,\ldots,x_n)$ specifying a division of the n goods. Once an agreement is reached, the agreed-upon allocations of the goods are implemented. This procedure permits agents to exploit the benefits of trading concessions on different issues.

The players' preferences are modelled by assuming that each player ag_i discounts future payoffs at some given rate δ_i^t, $0 < \delta_i^t < 1$, (δ_i^t is referred to as the discount factor and the preferences as time preferences with a constant discount rate). The cost of bargaining derives from the delay in consumption implied by a rejection of an offer. Practically speaking, the justification for this form of preferences takes into account the fact that money today can be used to make money tomorrow. Let U_i be the payoff function of ag_i. For simplicity and tractability, we assume that U_i is separable in all their arguments and that the per-period delay costs are the same for all issues:

$$U_i(x_1,\ldots,x_n,t) = \delta_i^{(t-1)} \ \sum_{l=1}^{n} w_{il} \ u_{il}(x_l)$$

where w_{il} is the weight of is_{il} and x_l denotes the share of ag_i for is_{il}. The component payoff function u_{il} for is_{il} is a continuous, strictly monotonic, and linear function. The distinguish feature of time preferences with a constant discount rate is the linearity of the function u_{il} [13]. The payoff of disagreement is normalized at 0 for both players.

2.2 Equilibrium Strategies

Game theory can provide sound design principles for computer scientists. Accordingly, this sub-section use game-theoretic techniques to define equilibrium strategies for the bargaining game of alternating offers (see, e.g., [13] for an in-depth description of the standard concept of equilibrium).

The negotiation process is modelled as an extensive game. For theoretical convenience, we consider the standard game-theoretic situation of two players completely informed about the various aspects of the game. The players are assumed to be rational, and each player knows that the other acts rationally. We also consider settings involving more than one issue. In particular, to reduce the complexity of the analysis, and without loss of generality, we consider a two-sided four-issue bargaining situation in which the players have different evaluations of the issues.

Two players are jointly endowed with a single unit of each of four goods, $\{X_1, \ldots, X_4\}$, and alternate proposals until they find an agreement. Each good is modelled as an interval $[0, 1]$ (or as a divisible pie of size 1). The players' preferences are as follows:

$$U_i = \delta_i^{(t-1)}\ (a\,x_1 + b\,x_2 + x_3 + x_4)$$
$$U_j = \delta_j^{(t-1)}\ [(1 - x_1) + (1 - x_2) + c\,(1 - x_3) + d\,(1 - x_4)]$$

where x_l and $(1 - x_l)$, $l = 1, \ldots, 4$, denote the shares of ag_i and ag_j for each pie, respectively. The parameters a, b, c, and d allow the marginal utilities of the players to differ across issues and players. We consider $a > b > 1$ and $d > c > 1$, *i.e.*, ag_i places greater emphasis on goods X_1 and X_2 while ag_j values goods X_3 and X_4 more. Also, we consider that δ_i and δ_j are close to 1 and the parameters a, b, c, and d are close to one another. Let $p_{j\to i}^{t-1}$ and $p_{i\to j}^{t}$ denote the offers that ag_j proposes to ag_i in period $t-1$ and ag_i proposes to ag_j in period t, respectively. Consider the following strategies:

$$str_i^* = \begin{cases} \text{offer } (1,\ 1,\ x_{i3}^*,\ 0) & \text{if } ag_i\text{'s turn} \\ \text{if } U_i(p_{j\to i}^{t-1}) \geq U_i^* \quad \text{accept else reject} & \text{if } ag_j\text{'s turn} \end{cases}$$

$$str_j^* = \begin{cases} \text{offer } (1,\ x_{j2}^*,\ 0,\ 0) & \text{if } ag_j\text{'s turn} \\ \text{if } U_j(p_{i\to j}^{t}) \geq U_j^* \quad \text{accept else reject} & \text{if } ag_i\text{'s turn} \end{cases}$$

where $U_i^* = U_i(1, x_{j2}^*, 0, 0)$, $U_j^* = U_j(1, 1, x_{i3}^*, 0)$, and the shares are the following: $x_{i3}^* = \frac{\delta_i\delta_j(a+b) - \delta_j(a+b+bc+bd) + bc + bd}{bc - \delta_i\delta_j}$ and $x_{j2}^* = \frac{\delta_i(\delta_i\delta_j(a+b) - \delta_j(a+b+bc+bd) + bc + bd) + (bc - \delta_i\delta_j)(a\delta_i + b\delta_i - a)}{b(bc - \delta_i\delta_j)}$.

Remark 1. For the two-sided four-issue bargaining game of alternating offers with an infinite horizon, in which the players' preferences are as described above, the pair of strategies (str_i^*, str_j^*) form an equilibrium. The outcome is the following:

$$x_1^* = 1, \quad x_2^* = 1, \quad x_3^* = \frac{\delta_i\delta_j(a+b) - \delta_j(a+b+bc+bd) + bc + bd}{bc - \delta_i\delta_j}, \quad x_4^* = 0$$

Agreement is immediately reached with no delay. The outcome is Pareto optimal.

The formal proof is presented in [10]. In short, note that the familiar necessary conditions for equilibrium are that ag_i is indifferent between waiting one period to have its offer accepted and accepting ag_j's offer immediately, and that ag_j is indifferent between waiting one period to have its offer accepted and accepting ag_i's offer immediately. Let $\mathbf{x}_i^* = (x_{i1}^*, \ldots, x_{i4}^*)$ and $\mathbf{x}_j^* = (x_{j1}^*, \ldots, x_{j4}^*)$ be the equilibrium proposals of ag_i and ag_j, respectively. The problem for ag_i is stated as follows:

maximize:

$$U_i(x_1,\ldots,x_4,t) = \delta_i^{(t-1)}\ (ax_1 + bx_2 + x_3 + x_4)$$

subject to:

$$(1-x_{i1}^*)+(1-x_{i2}^*)+c(1-x_{i3}^*)+d(1-x_{i4}^*) = \delta_j[(1-x_{j1}^*)+(1-x_{j2}^*)+c(1-x_{j3}^*)+d(1-x_{j4}^*)]$$

$$0 \le x_{il}^* \le 1, \quad 0 \le x_{jl}^* \le 1, \quad \text{for} \quad l=1,\ldots,4$$

The problem for ag_j is stated in a similar way and is omitted. Solving both maximization problems yields the outcome specified in the statement of the Remark. In the limit, letting $\delta_i \to 1$ and $\delta_j \to 1$, the outcome of the equilibrium is $(1,1,0,0)$. This outcome is on the Pareto frontier and corresponds to the utility pair $(a+b,\ c+d)$.

At this stage, we hasten to add an explanatory note. For convenience, this sub-section has considered a simple two-sided four-issue bargaining situation in which the players have different evaluations of the issues. Nevertheless, more complex situations, involving different number of issues, can be handled similarly (but see [10]).

2.3 Concession and Problem Solving Strategies

Behavioural negotiation theory can provide rules-of-thumb to agent designers. The danger is that the designers may not be fully aware of the circumstances to which human practice is adapted, and hence use rules that can be badly exploited by new agents. Nevertheless, an increasing number of researchers consider that human practice is important to automated negotiation (see, e.g., [1,5,12]).

Now, the last subsection has considered two fully informed agents and used game-theoretic techniques to define equilibrium strategies. The agents were able to settle for the outcome that maximizes their benefit. Yet, the assumption of complete information is of limited use to the designers of agents. In practice, agents have private information. Also, simple casual observation reveals the existence of concessions and long periods of disagreement in many actual negotiations. Accordingly, this subsection considers two incompletely informed agents about the various aspects of the bargaining game and formalizes relevant strategies studied in the social sciences.

Negotiation strategies can reflect a variety of behaviours and lead to strikingly different outcomes. However, the following two fundamental groups of strategies are commonly discussed in the behavioural negotiation literature [14,15]:

1. *concession making* – negotiators who employ strategies in this group reduce their aspirations to accommodate the opponent;
2. *problem solving* – negotiators maintain their aspirations and try to find ways of reconciling them with the aspirations of the opponent.

Two explanatory and cautionary notes are in order here. First, most strategies are implemented through a variety of tactics. The line between strategies and tactics often seems indistinct, but one major difference is that of scope. Tactics are short-term moves designed to enact or pursue broad (high-level) strategies [7]. Second, most strategies are only informally discussed in the behavioural literature – they are not formalized, as typically happens in the game-theoretic literature.

Concession making behaviour aims at partially or totally accommodating the other party. Consider two incompletely informed agents bargaining over n distinct issues $\{is_1, \ldots, is_n\}$. For convenience, each issue is_l is modelled as an interval $[min_l, max_l]$. The agents' preferences are as defined in subsection 2.1. The opening stance and the pattern of concessions are two central elements of negotiation. Three different opening positions (extreme, reasonable and modest) and three levels of concession magnitude (large, moderate and small) are commonly discussed in the behavioural literature [7]. They can lead to a number of concession strategies, notably:

1. *starting high and conceding slowly* – negotiators adopt an optimistic opening attitude and make successive small concessions;
2. *starting reasonable and conceding moderately* – negotiators adopt a realistic opening attitude and make successive moderate concessions.

Let $p^{t-1}_{j\to i}$ be the offer that ag_j has proposed to ag_i in period $t-1$. Likewise, let $p^t_{i\to j}$ be the offer that ag_i is ready to propose in the next time period t. The formal definition of a generic concession strategy follows.

Definition 1. *Let $ag_i \in Ag$ be a negotiating agent. A concession strategy for ag_i is a function that specifies either the tactic to apply at the beginning of negotiation or the tactic that defines the concessions to be made during the course of negotiation:*

$$conc \overset{def}{=} \begin{cases} apply\ tact_i^1 & if\ ag_i\text{'s turn and } t=1 \\ apply\ tact_i^t & if\ ag_i\text{'s turn and } t>1 \\ if\ U_i(p^{t-1}_{j\to i}) \geq U_i(p^t_{i\to j}) \quad accept\ \ else\ \ reject & if\ ag_j\text{'s turn} \end{cases}$$

where $tact_i^1$ is an opening negotiation tactic and $tact_i^t$ is a concession tactic. ∎

The two aforementioned concession strategies are defined by considering different tactics. For instance, the "starting high and conceding slowly" strategy is defined by: "$tact_i^1 = starting\_optimistic$" and "$tact_i^t = tough$" (but see below).

Problem solving behaviour aims at finding agreements that appeal to all sides, both individually and collectively. This behaviour can take several forms, notably logrolling – negotiators agree to trade-off among the issues under consideration so that each party concedes on issues that are of low priority to itself and high priority to the other party [14,15]. Effective logrolling requires information about the two parties' priorities so that concessions can be matched up. This information is not always easy to get. The main reason for this is that negotiators often try to conceal their priorities for fear that they will be forced to concede on issues of lesser importance to themselves without receiving any repayment [14]. Despite this, research evidence indicates that it is often not detrimental for negotiators to disclose information that can reveal their priorities – a simple rank order of the issues does not put negotiators at a strategic disadvantage [15]. Hence, we consider that negotiators willingly disclose information that can help to identify their priorities (e.g., their interests). The formal definition of a generic logrolling strategy follows.

Definition 2. *Let $ag_i \in Ag$ be a negotiating agent and $ag_j \in Ag$ be its opponent. Let $Agenda$ denote the negotiating agenda, $Agenda^{\oplus}$ the subset of the agenda containing* *the issues of high priority for ag_i (and low priority for ag_j), and $Agenda^{\ominus}$ the subset of the agenda containing the issues of low priority for ag_i (and high priority for ag_j). A logrolling strategy for ag_i is a function that specifies either the tactic to apply at the beginning of negotiation or the tactics to make trade-offs during the course of negotiation:*

$$log \overset{def}{=} \begin{cases} \textit{apply } tact_i^1 & \textit{if } ag_i\textit{'s turn and } t=1 \\ \textit{apply } tact_i^{t\oplus} \textit{and } tact_i^{t\ominus} & \textit{if } ag_i\textit{'s turn and } t>1 \\ \textit{if } U_i(p_{j\to i}^{t-1}) \geq U_i(p_{i\to j}^{t}) \quad \textit{accept else reject} & \textit{if } ag_j\textit{'s turn} \end{cases}$$

where $tact_i^1$ is an opening negotiation tactic, $tact_i^{t\oplus}$ is a concession tactic (to apply to the issues on $Agenda^{\oplus}$), and $tact_i^{t\ominus}$ is another concession tactic (to apply to the issues on $Agenda^{\ominus}$). ∎

A number of logrolling strategies can be defined simply by considering different tactics. For instance, a strategy that specifies an optimistic opening attitude, followed by null concessions on issues on $Agenda^{\oplus}$, and small concessions on issues on $Agenda^{\ominus}$, is defined by: "$tact_i^1 = starting\_optimistic$", "$tact_i^{t\oplus} = stalemate$", and "$tact_i^{t\ominus} = tough$". Similarly, a strategy that specifies a realistic opening attitude, followed by null concessions on issues on $Agenda^{\oplus}$, and large concessions on issues on $Agenda^{\ominus}$, is defined by: "$tact_i^1 = starting\_realistic$", "$tact_i^{t\oplus} = stalemate$", and "$tact_i^{t\ominus} = soft$" (but see below).

At this stage, it is worth making the point that logrolling is a major route – though not the only route – to the development of mutually superior solutions (*i.e.*, solutions thar are better for all parties). In fact, the host of existing problem solving strategies includes expanding the "pie", nonspecific compensation, cost cutting, and bridging. These strategies are implemented by different sets of tactics and require progressively more information about the other parties (but see [14]).

Opening negotiation tactics are functions that specify the initial values for each issue is_l at stake. The following three tactics are commonly discussed in the behavioural literature [7]:

1. *starting optimistic* – specifies a value far from the target point;
2. *starting realistic* – specifies a value close to the target point;
3. *starting pessimistic* – specifies a value close to the limit.

The definition of the tactic "starting realistic" follows (the definition of the other two tactics is essentially identical, and is omitted).

Definition 3. *Let $ag_i \in Ag$ be a negotiating agent and $is_l \in Agenda$ a negotiation issue. Let trg_{il} be the target point of ag_i for is_l. The tactic starting realistic for ag_i is a function that takes is_l and trg_{il} as input and returns the initial value $v[is_l]_i^1$ of is_l:*

$$starting\_realistic(is_l, trg_{il}) = v[is_l]_i^1$$

where $v[is_l]_i^1 \in [\,trg_{il} - \epsilon,\ trg_{il} + \epsilon]$ and $\epsilon > 0$ is small. ∎

Concession tactics are functions that compute new values for each issue is_l. The following five tactics are commonly discussed in the literature [7]:

1. *stalemate* – models a null concession on is_l;
2. *tough* – models a small concession on is_l;
3. *moderate* – models a moderate concession on is_l;
4. *soft* – models a large concession on is_l;
5. *accommodate* – models a complete concession on is_l.

The definition of a generic concession tactic follows (without loss of generality, we consider that ag_i wants to maximize is_l).

Definition 4. *Let $ag_i \in Ag$ be a negotiating agent, $is_l \in Agenda$ a negotiation issue, and lim_{il} the limit of is_l. Let $v[is_l]_i^t$ be the value of is_l offered by ag_i at period t. A concession tactic for ag_i is a function that takes $v[is_l]_i^t$, lim_{il} and the concession factor $Cf \in [0,1]$ as input and returns the new value $v[is_l]_i^{t+2}$ of is_l:*

$$concession\_tactic(v[is_l]_i^t, lim_{il}, Cf) = v[is_l]_i^{t+2}$$

where $v[is_l]_i^{t+2} = v[is_l]_i^t - Cf\,(v[is_l]_i^t - lim_{il})$. ∎

The five tactics are defined by considering different values for Cf. In particular, the stalemate tactic by $Cf = 0$, the accommodate tactic by $Cf = 1$, and the other three tactics by different ranges of values for Cf (e.g., the tough tactic by $Cf \in]0.00, 0.05]$, the moderate tactic by $Cf \in]0.05, 0.10]$, and the soft tactic by $Cf \in]0.10, 0.15]$).

3 Experimental Analysis: Preliminary Report

The experimental method is controlled experimentation [2]. The experiment involves three types of agents: (i) competitive or individualistic agents, (ii) cooperative agents, and (ii) strategically creative agents. Competitive agents show a strong interest in achieving only their own outcomes – getting this deal, winning this negotiation – and pursue the starting high and conceding slowly strategy (see definition 1 and the comments above). Cooperative agents are concerned with both their own and the other's outcomes – building, preserving, or enhancing a good relationship with the other party – and pursue a logrolling strategy, which specifies a realistic opening attitude, null concessions on issues on $Agenda^{\oplus}$, and small concessions on issues on $Agenda^{\ominus}$ (see definition 2, above). Similarly, strategically creative agents pursue a logrolling strategy specifying a realistic opening attitude, followed by null concessions on issues on $Agenda^{\oplus}$, and large concessions on issues on $Agenda^{\ominus}$ (see again definition 2, and the comments above). This strategy permits agents to better exploit the differences in the valuations of the issues.

The experimental system consists of two autonomous agents and a simulated environment. Let $Ag = \{ag_s, ag_b\}$ be the set of agents. The agent ag_s plays the role of a seller and the agent ag_b the role of a buyer. The agents negotiate the price of four generic commodities. The seller places greater emphasis on the price of the first two commodities, while the buyer values the last two commodities more. The experimental hypotheses are related to the quality and the cost of bargaining (a dyad is composed by a seller agent and a buyer agent):

Hypothesis 1. Dyads whose seller agent is cooperatively oriented and pursue a problem solving strategy (a logrolling strategy) will reach higher quality outcomes (higher joint benefits) than dyads whose seller agent is individualistically oriented and pursue a concession strategy (a "starting high and conceding slowly" strategy).

Hypothesis 2. Cooperatively oriented dyads composed by a creative seller agent pursuing the logrolling strategy that better exploits the differences in the valuation of the issues (and thus, involving large or complete concessions on low-priority issues) will reach higher quality outcomes (higher joint benefits) than cooperatively oriented dyads composed by a cooperative seller agent pursuing a logrolling strategy involving smaller concessions on low-priority issues;

Hypothesis 3. The cost of bargaining (the time spent in negotiation) will be lower in individualistically oriented dyads composed by a competitive seller agent than in cooperatively oriented dyads composed by a cooperative or a creative seller agent.

The independent variable is the bargaining orientation of the seller agent (manipulated by assigning a specific strategy to this agent). This variable has three levels, namely the three strategies mentioned earlier. The first dependent variable is the joint benefit provided by the final agreement, $i.e.$, the sum of the two agents' benefits in the final agreement. The second dependent variable is the time spent in negotiation. This variable is measured in terms of the total number of offers exchanged by the agents.

The experiment involves three groups of trials. Each group corresponds to a level of the independent variable. A trial is a single run of the experimental system and involves a bargaining session. Trials of the same group will, in general, differ from one another. The detailed experimental procedure is a follows:

1. for each group of trials, the experimenter manipulates the independent variable, $i.e.$, assigns a strategy to the seller agent;
2. for each trial in each group, the experimenter randomly determines the agent that starts the bidding process and the orientation of the buyer agent, $i.e.$, its strategy (from a library containing several concession and problem solving strategies);
3. for all trials of each group, the experimenter measures the dependent variables and computes averages on the measures taken.

4 Related Work

AI researchers have paid a great deal of attention to automated negotiation over the past decade and a number of prominent models have been proposed in the literature (see, e.g., [4,12]. The majority of models primarily use either game-theoretic techniques or methods from the social sciences as a basis to develop autonomous negotiating agents, and largely ignore the integration of the results from both research areas. Yet, the task of designing and implementing negotiating agents is usually understood as involving the consideration of insights from multiple relevant research areas to integrate different perspectives on negotiation [1]. In particular, game-theoretic (strategic) and behavioural negotiation theories can mutually reinforce each other and lead to more comprehensive and richer models. Accordingly, this paper uses both game-theoretic techniques and methods from the social sciences as a basis to develop negotiating agents.

5 Conclusion

This paper has used both game-theoretic techniques and methods from the social sciences as a basis to develop autonomous agents with negotiation competence. On the one hand, it has considered two fully informed agents about the various aspects of the bargaining game of alternating offers and has employed game-theoretic techniques to define equilibrium strategies. On the other hand, it has considered two incompletely informed agents and has formalized a set of concession and problem solving strategies based on rules-of-thumb distilled from human behavioural practice. It has also laid the foundation for performing an experiment to investigate the behaviour of these strategies, paying particular attention to the quality and the cost of bargaining.

Autonomous agents able to negotiate under both complete and incomplete information are currently being developed using the JAVA programming language. Our aim for the future is to perform a set of inter-related experiments to empirically evaluate the key components of the agents. Each experiment will lay the foundation for subsequent experimental work.

References

1. Bichler, M., Kersten, G., Strecker, S.: Towards a Structured Design of Electronic Negotiations. Group Decision and Negotiation 12, 311–335 (2003)
2. Cohen, P.: Empirical Methods for Artificial Intelligence. MIT Press, Cambridge (1995)
3. Fatima, S., Wooldridge, M., Jennings, N.: A Comparative Study of Game Theoretic and Evolutionary Models of Bargaining for Software Agents. Artificial Intelligence Review 23, 185–203 (2005)
4. Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Wooldridge, M., Sierra, C.: Automated Negotiation: Prospects, Methods and Challenges. Group Decision and Negotiation 10, 199–215 (2001)
5. Kraus, S.: Negotiation and Cooperation in Multi-Agent Environment. Artificial Intelligence 94, 79–98 (1997)
6. Kraus, S.: Strategic Negotiation in Multi-Agent Environments. MIT Press, Cambridge (2001)
7. Lewicki, R., Barry, B., Saunders, D., Minton, J.: Negotiation. McGraw Hill, New York (2003)
8. Lopes, F., Mamede, N., Novais, A.Q., Coelho, H.: A Negotiation Model for Autonomous Computational Agents: Formal Description and Empirical Evaluation. Journal of Intelligent&Fuzzy Systems 12, 195–212 (2002)
9. Lopes, F., Mamede, N., Novais, A.Q., Coelho, H.: Negotiation Strategies for Autonomous Computational Agents. In: ECAI 2004, pp. 38–42. IOS Press, Amsterdam (2004)
10. Lopes, F., Fatima, S., Wooldridge, M.: Pre-Negotiation and Impasse in Automated Negotiation. submitted to Group Dec. and Neg. (2007)
11. Lopes, F., Novais, A.Q., Coelho, H.: Towards an Interdisciplinary Framework for Automated Negotiation. In: Darzentas, J., Vouros, G.A., Vosinakis, S., Arnellos, A. (eds.) SETN 2008. LNCS, vol. 5138, pp. 81–91. Springer, Heidelberg (2008)
12. Lopes, F., Wooldridge, M., Novais, A.Q.: Negotiation Among Autonomous Computational Agents: Principles, Analysis and Challenges. Artificial Intelligence Review (in press, 2009)
13. Osborne, M., Rubinstein, A.: Bargaining and Markets. Academic Press, London (1990)
14. Pruitt, D., Kim, S.: Social Conflict: Escalation, Stalemate, and Settlement. McGraw Hill, New York (2004)
15. Thompson, L.: The Mind and Heart of the Negotiator. Prentice-Hall, Englewood Cliffs (2005)

A Decentralized Scheduling Policy for a Dynamically Reconfigurable Production System

Stefano Giordani[1], Marin Lujak[1], and Francesco Martinelli[2]

[1] Dip. Ingegneria dell'Impresa - University of Rome "Tor Vergata", Italy
[2] Dip. Informatica Sistemi e Produzione - University of Rome "Tor Vergata", Italy

Abstract. In this paper, the static layout of a traditional multi-machine factory producing a set of distinct goods is integrated with a set of mobile production units - robots. The robots dynamically change their work position to increment the product rate of the different typologies of products in respect to the fluctuations of the demands and production costs during a given time horizon. Assuming that the planning time horizon is subdivided into a finite number of time periods, this particularly flexible layout requires the definition and the solution of a complex scheduling problem, involving for each period of the planning time horizon, the determination of the position of the robots, i.e., the assignment to the respective tasks in order to minimize production costs given the product demand rates during the planning time horizon.

We propose a decentralized multi-agent system (MAS) scheduling model with as many agents as there are the tasks in the system, plus a resource (robot) owner which assigns the robots to the tasks in each time period on the basis of the requests coming from the competing task agents. The MAS model is coupled with an iterative auction based negotiation protocol to coordinate the agents' decisions. The resource prices are updated using a strategy inspired by the subgradient technique used in the Lagrangian relaxation approach. To measure the effectiveness of the results, the same are evaluated in respect to that of the benchmark centralized model.

Keywords: Dynamic scheduling, multi-agent system, negotiation.

1 Introduction

Different solutions have been adopted through the years in the manufacturing domain in order to speed up the system's response to a fluctuating external demand. Some of the approaches to this problem include Flexible Manufacturing Systems (see, e.g., Huang and Chen, 1986), Group Technology (see, e.g., Selim et al., 1998), Holonic Manufacturing (see, e.g., Christensen, 1994), and Agile Productions Systems. Design and integration of robot assistance in manufacturing is a more recent research topic which also moves in this direction (see, e.g., Helms et al., 2002). Referring to the latter, we propose a scheduling solution to the problem where some production resources (machines) are assumed to be mobile units and can perform several different manufacturing tasks in the plant.

In more detail, we consider a manufacturing plant producing a set of distinct products where the tasks (i.e., assemblies of specific products) are executed on different fixed locations on the shop floor and require (additional) production resources represented by mobile manufacturing units, i.e., robots. The robots are assigned to the tasks where the production is executed, and dynamically change their location in respect to the tasks' local change in demand. A task can be seen as a non-preemptive set of manufacturing operations and all the tasks are assumed to be independent (or parallel). This for example occurs in the final stage of assembly process where the products, after the execution of the operations corresponding to a certain final assembly task on them, get distributed to the warehouses or clients, and, therefore, leave the system. The planning time horizon is assumed to be subdivided into a finite number of time periods. The problem we address consists in dynamically assigning the robots to the tasks, and minimizing production costs given the product demand rates during the planning time horizon.

From the organizational point of view, it is possible to identify two kinds of scenarios in an industrial factory: 1) the management of the plant is centralized, i.e., all the decisions are made by a single decision maker managing the whole system information; 2) the management of the plant is decentralized, i.e., the decisions are taken only using the local information by the set of local decision makers (agents) which are possibly in competition. The drawbacks of centralized systems are a lack of reliability since a critical point of failure in the network is a single central agent itself, and the elevated system's complexity as the system increases in size (see, e.g. Kurose and Simha, 1989, Sycara, 1998). In this paper, the latter, decentralized approach is adopted e.g. in the case when there is a significant number of production machines and the centralized system can not cope with the system's complexity in the real time. We analyze this scenario with a set of (task) agents representing the tasks which compete for the robots, and the resource (robot) owner, being in charge of managing the negotiation among task agents, and of assigning the robots to the tasks. This approach helps in building a highly scalable system since the decision-making process is distributed across all the fixed production machines and the resource owner, and therefore the decisions about the production lie in the hands of each one of them by locally maximizing their individual utility and profit through the process of multi-agent interaction and negotiation. Furthermore, market price mechanisms for resource allocation are by their nature distributed with each auction being held within certain limited dimensions. The most important advantages of this approach are modularity, regular interactions, incrementality, analyzability, and incentive engineering (see, e.g., Clearwater, 1996).

For this paper's scenario, a discrete time decentralized multi-agent system model coupled with an iterative auction based negotiation protocol (see, e.g., Kutanoglu and Wu, 2006, Schneider et al., 2005, and Robin, 1991) is proposed.

At the beginning of each iteration, the robot owner sets the robot hiring price to a given value for each time period. Each task agent, using this information together with its local information on demands and production costs of its specific product, performs its local optimization and makes a request to the robot owner indicating the number of robots it wants to hire for each time period. The robot owner collects the task agent requests, finds a feasible robot assignment to tasks for each time period and, on the basis of the (positive or negative) deviation between the number of the total robot requests and the number of the available robots, changes the robot hiring prices for each time period so that the effective prices reach as much as possible the equilibrium between the number of the robots in the system and the number of the requested ones. In particular, an approach based on the subgradient technique used in Lagrangian relaxation is adopted (see, e.g., Chen et al., 1998, and Barahona and Anbil, 2000).

We experiment the proposed model considering highly variable demands in the planning time horizon: an ARMA model is used to model each task's demand fluctuations in the considered time horizon. To measure the effectiveness of the approach, a discrete time centralized model is used as a benchmark for the purpose of evaluating the fluctuation of the social welfare (see, e.g., Chevaleyre et al., 2006, Chevaleyre et al., 2004) of the task agents in the decentralized scenario.

The remainder of the paper is organized as follows. In Section 2 we review some of the economic models used in MAS negotiation in resource allocation. In section 3 we describe the decentralized scheduling problem, and formulate the optimization problem addressed by each task agent. In Section 4 we describe the proposed negotiation process among agents. Furthermore, we give more details on the dynamic adjustment in subgradient optimization used in the auction algorithm, and on the evaluation of the quality of the robot assignment. Finally, in Section 5, we experiment the proposed decentralized model and compare its performance with that of the centralized scenario.

2 Negotiation Models in MAS Resource Allocation

For the multi-agent interaction and negotiation, there are several applicable economic models which might work well in this case (see, e.g., Kraus, 2001, Chevaleyre et al., 2006, Buyya et al., 2002). They are: Commodity Market model, where more resource owners specify their service price and charge users according to the amount of resource they consume; Posted price model, similar to the commodity market model, except that it publishes special offers to attract consumers to establish market share or motivate users to use cheaper offers. In this case there is no need for brokers to negotiate directly with clients for price, since the posted prices are generally cheaper than regular prices; Bargaining model where resource brokers bargain with clients for lower access price and higher usage duration; Tender/Contract-net model, which original version lacks a formal model for making bidding and awarding decisions. The Contract-net framework was extended to work among self-interested, computationally limited agents which decisions are based on marginal approximations of cost calculations. The advantage of this model is that if the selected contractor is unable to deliver a satisfactory service, the brokers can seek services of other contractors. This protocol has certain disadvantages. A task might be awarded to a less capable contractor if a more capable contractor is busy at award time.

In Auction Model, which is the approach chosen in this paper, one-to-many negotiation is supported, between a service provider (seller) who offers the goods or services for bids and then sells the item(s) to one winning out of many bidders (buyers) who give the bids to the seller. In the iterative version, the buyers can update the bids and the provider in respect to the offers updates the offered sale price. There are several variations on the basic auction form, including time limits, minimum or maximum limits on bid prices, and special rules for determining the winning bidder(s) and sale price(s). Auctions basically use market forces to negotiate a clearing price for the service and can be conducted as open or closed depending on whether they allow back-and-forth offers and counter offers (see, e.g., Vulkan and Jennings, 2000).

3 Decentralized Scheduling Problem

A manufacturing system producing P products is considered. The production is realized by the execution of different and independent (i.e. parallel) tasks, one for each product, requiring (additional) resources represented by a set of N identical mobile robots, all at disposal of the tasks. Given a finite production time horizon of T time periods, we consider the following decentralized discrete time production scheduling problem, where each task agent requests robots from the robot owner for each time period and schedules the production based on product demands and production costs, and the robot owner assigns the robots to the tasks for each unit time period on the basis of their requests.

The agent representing the task i knows, for each time period k, product demand $d_i(k)$, robot maximum production rate $r_i(k)$ (production capacity), unitary manufacturing cost $c_i(k)$, unitary holding cost $h_i(k)$, unitary backlog cost $b_i(k)$, and unitary robot hiring cost $\rho_i(k)$, all nonnegative. For each time period, besides the robot hiring cost, a task has to pay also the cost of displacement of the additionally assigned robots in the time period with respect to the robots assigned to it in the previous time period. The same holds for the end of the planning time horizon, T, when the tasks with assigned robots pay the costs of the robot displacement to the robot depot. It is reasonable to assume that the robot's displacement time is negligible in respect to the length of the time period, Δt, and that the robot displacement cost does not depend on the locations of the tasks in the factory. This stands for the physical layout of the plant which is circular with radius R, as shown in Fig. 1. Therefore, let $\delta_i(k) = \delta$ be the cost of displacement of an additional robot to the location of task i at the beginning of period k (with $k = 1, \ldots, T$), and let $\delta_i(T+1) = \delta$ be the cost of displacement of a robot to the robot depot at the end of the planning time horizon. For each time period k, the decision variables controlled by the agent of task i are:

- $n_i(k) \geq 0$: required number of robots (robot request);
- $\Delta n_i(k) \geq 0$: number of additional robots in the present time period in respect to the one required in the period before;
- $u_i(k) \geq 0$: production rate.

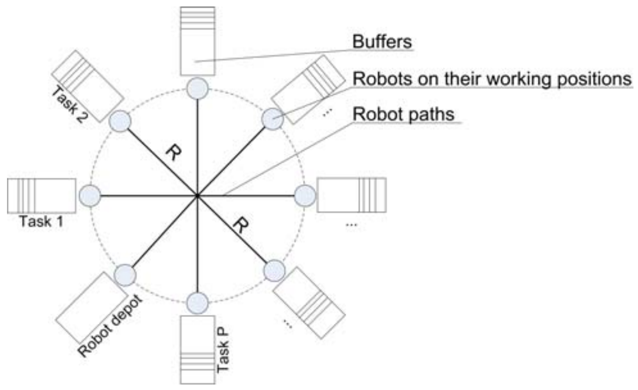


Fig. 1. Circular shop-floor model

Production can be anticipated or delayed in respect to the product demands resulting in holding and backlog costs, respectively. Each task i is associated with a buffer which content $x_i(k)$ at the end of time period k can be positive (if a stock of completed product items is present in the buffer) or negative (if a backlog of demands for the product realized with task i is in the queue). Using a standard notation, we denote $x_i^+(k) := \max\{x_i(k), 0\}$ as the stock level, and $x_i^-(k) := \max\{-x_i(k), 0\}$ as the backlog level at the end of time period k. Notice that, for each time period k, and task (product) i, only one of $x_i^+(k)$ and $x_i^-(k)$ can be different from 0 (i.e., $x_i^+(k) \cdot x_i^-(k) = 0\ ,\ \ \forall k\ ,\ \ \forall i$).

The local optimization problem addressed by every task agent i includes $x_i^+(k)$ and $x_i^-(k)$ as additional variables, for each time period k. Those variables, together with the production rates $u_i(k)$, are assumed to be continuous according to a fluid approximation. Given the above parameters and variables, the local optimization problem, P_i, addressed by each task agent i can be formulated as follows.

(P_i) :

$$\min z_i = \sum_{k=1}^{T} \left[h_i(k)x_i^+(k) + b_i(k)x_i^-(k) + c_i(k)u_i(k)\right] +$$

$$\sum_{k=1}^{T} [\rho_i(k)n_i(k) + \delta_i(k)\Delta n_i(k)] + \delta_i(T+1)n_i(T)\ , \tag{1}$$

subject to:

$$x_i^+(k) - x_i^-(k) = x_i^+(k-1) - x_i^-(k-1) + \Delta t[u_i(k) - d_i(k)],\ \ k = 1, ..., T \tag{2}$$

$$u_i(k) \le r_i(k)\, n_i(k),\ \ k = 1, ..., T; \tag{3}$$

$$n_i(k) - n_i(k-1) \le \Delta n_i(k),\ \ k = 1, ..., T; \tag{4}$$

$$u_i(k), x_i^+(k), x_i^-(k), \Delta n_i(k) \geq 0, \ \ k = 1, ..., T; \tag{5}$$

$$0 \leq n_i(k) \leq N \ \text{ and integer}, \ \ k = 1, ..., T. \tag{6}$$

The values of $x_i^+(0)$, $x_i^-(0)$, and $n_i(0)$ represent the initial conditions, i.e. the stock level, the backlog level, and the number of preassigned robots to task i respectively, at the beginning of the planning time horizon. These values are assumed to be equal to zero. Constraints (2) are the mass balance constraints among product demand $\Delta t \cdot d_i(k)$, stock level $x_i^+(k)$, backlog level $x_i^-(k)$, and production level $\Delta t \cdot u_i(k)$, for each time period k. W.l.o.g., assuming that, for each time period k and for each task i, the values of $h_i(k)$ and $b_i(k)$ are positive, the constraint $x_i^+(k) \cdot x_i^-(k) = 0$, is implicitly satisfied by the optimal solution, and, hence, omitted in the formulation. Constraints (3) limit the production rate $u_i(k)$ to be not greater than the production capacity $r_i \cdot n_i(k)$. Constraints (4) together with the (non-negative) robot displacement cost allow to find the value $\Delta n_i(k)$ of additionally assigned robots: note that there is no need to define this variable as an integer in the formulation.

For each task agent i, let $n_i^*(k)$ be the number of required robots at time period k, in the optimal solution of problem P_i, and let z_i^* be the optimal solution of P_i. Each task agent (decision maker) finds such a solution taking into account only its local objective and constraints, but shares with the other task agents limited amount of available robots (resources). Therefore, these local decisions can be implemented only if the following global constraints are satisfied.

$$\sum_{i=1}^{P} n_i^*(k) \leq N, \ \ k = 1, ..., T. \tag{7}$$

In general, this is not the case, and, hence, a negotiation process must be implemented among the task agents to come up with a set of local solutions that together satisfy also Constraints (7). We assume that this process is supervised by another decision maker, i.e. the robot owner, that assigns the robots to the tasks for each time period on the basis of their requests, guaranteeing the fulfilment of the constraint on the limited robot amount. In the next section, we provide a model for such a negotiation.

4 Negotiation Process

The negotiation among the tasks is modeled by an iterative auction process controlled by the robot owner which is in charge of coordination of the negotiation. At each iteration:

Step 1. The robot owner (*auctioneer*) communicates to the task agents (*bidders*) current prices of the robots in the T time periods;

Step 2. Each task agent based on the local utility function of its local objective, the constraints, and the current robot prices, determines the robot requests (a *bid*) for the T time periods maximizing its utility function, and communicates its robot requests to the robot owner;

Step 3. Based on the bids received from the task agents, the robot owner allocates the robots to the tasks, for each time period, maximizing its own utility function;

Step 4. In order to reduce the possible conflicts among tasks generated by their bids (i.e., the violation of Constraints (7)), or to stimulate the usage of the robots, the robot owner updates the robot prices.

The iteration process repeats until a certain (halting) condition is reached (e.g., a maximum number of iterations have been performed, or the best robot allocation found is sufficiently good from the agents' point of view). At the end, the best robot allocation is retrieved.

Let $B_i = \{n_i(k) : k = 1, \ldots, T\}$ be a bid of task agent i, and let $\Lambda = \{\lambda(k) : k = 1, \ldots, T\}$ be the (current) set of the robot prices fixed by the robot owner (each one for each time period). The utility function $U_i(B_i, \Lambda)$ of agent i is the opposite of the total production cost during the planning time horizon:

$$U_i(B_i, \Lambda) = -z_i^*(B_i) - p(B_i, \Lambda). \tag{8}$$

In this expression, $z_i^*(B_i)$ is the (minimum) production cost for task i with fixed numbers of assigned robots $n_i(k)$, with $k = 1, \ldots, T$, according to bid B_i, and $p(B_i, \Lambda) = \sum_{k=1}^{T} \lambda(k) n_i(k)$ is the (additional) robot cost to be payed to the robot owner. Note that, $z_i^*(B_i)$ is the optimal solution of the local optimization problem P_i with fixed values of $n_i(k)$.

In Step 2, agent i finds the best bid B_i^*, i.e., the bid that maximizes its utility function $U_i(B_i, \Lambda)$. Let $U_i^*(\Lambda)$ be its maximum value for a given set Λ of robot prices. Finding B_i^*, and hence determining $U_i^*(\Lambda)$, corresponds to solving the problem P_i with $\lambda(k)$ added to the robot hiring cost $\rho_i(k)$ in the objective function (1).

In Step 3, the resource owner receives bid B_i^* from each task i, and assigns robots to tasks maximizing its utility function taking into account the received bids. Denoting with $\nu_i(k)$ the number of robots assigned to task i in time period k, the utility function $R(\nu, \Lambda)$ of the resource owner is the total profit obtained from the robot assignment, and, hence, it is

$$R(\nu, \Lambda) = \sum_{k=1}^{T} \lambda(k) \sum_{i=1}^{P} \nu_i(k). \tag{9}$$

Since $\nu_i(k)$ cannot be greater than robot request (bid) $n_i^*(k)$ of task agent i and $\sum_{i=1}^{P} \nu_i(k)$ cannot be greater than N, for each $k = 1, ..., T$, the maximization of $R(\nu, \Lambda)$ can be easily obtained by assigning robots to tasks as follows.

For each time period k, order the tasks according to the non-increasing robot requests $n_i^*(k)$, and, following this task order, assign $\nu_i(k) = \min\{n_i^*(k), N'(k)\}$

robots to task i, where $N'(k)$ (with $0 \leq N'(k) \leq N$) is the number of non-assigned (yet available) robots in time period k.

The quality of the robot assignment is evaluated from the agents' point of view, measuring the *social welfare* $w(\nu)$ of the tasks related to a given robot assignment $\nu = \{\nu_i(k) | i = 1, \ldots, P, \ k = 1, \ldots, T\}$ as the (minimum) total production cost of the tasks with that robot assignment, that is,

$$w(\nu) = \sum_{i=1}^{P} z_i^*(\nu_i), \tag{10}$$

where $z_i^*(\nu_i)$ is the optimal solution of P_i with a given number of robots $n_i(k) = \nu_i(k)$ assigned to task i, for each time period k. Let w^* be the best (minimum) value of the social welfare found so far during the iterative auction process. It can be proved that the expression

$$w_L(\Lambda) = -\sum_{i=1}^{P} U_i^*(\Lambda) - N \sum_{k=1}^{T} \lambda(k), \tag{11}$$

is a valid lower bound on the best value of the agents' social welfare, for any vector Λ of non-negative robot prices $\lambda(k)$.

In Step 4 the resource owner updates the robot prices $\lambda(k)$, with $k = 1, ..., T$. This is done, considering the total requests' *deviation* $dev(k) = \sum_{i=1}^{P} n_i^*(k) - N$ of the total number of requested robots from the number N of available robots, and by increasing the current value of $\lambda(k)$ if $dev(k)$ is positive (i.e., $dev(k)$ is the *excess* of robot requirements), or decreasing it (at most to 0) if $dev(k)$ is negative (in this latter case $-dev(k)$ is the *deficit* of robot requirements). The value of the increase (decrease) of $\lambda(k)$ should be a non-decreasing function of the excess (deficit); moreover, it is a good practice that the auctioneer is more aggressive in the early iterations to get very quickly a good robot assignment, while smaller adjustments can be made in later iterations to refine the quality of the same. A possible choice for the price updating that goes in this direction is that of using an algorithm inspired by the subgradient technique used in Lagrangean relaxation, that experimentally guarantees the convergence of the Lagrangean dual (see, e.g., Held et al., 1974).

At iteration h of the auction process, let $dev^h(k)$ be the deviation of the robot requirements, and Λ^h be the vector of the robot prices $\lambda^h(k)$, with $k = 1, \ldots, T$. The new value of the robot price in time period k at iteration $h+1$ is:

$$\lambda^{h+1}(k) = \max \left\{ 0, \ \lambda^h(k) + \alpha^h \frac{w^* - w_L(\Lambda^h)}{\sum_{k=1}^{T} (dev^h(k))^2} dev^h(k) \right\}, \tag{12}$$

where α^h is a scalar controlling the aggressiveness of the auctioneer. Following analog practice in Lagrangean relaxation (see, e.g., Held et al., 1974) we start with value $\alpha_0 = 2$, and halve its value if the best task social welfare value found so far is not improved within a certain number of iterations.

In Step 1, the amount of information going from the robots' owner to the task agents is $O(P \cdot T \cdot \log(\lambda_{max}))$ bits, where λ_{max} is the maximum robot price. In Step 2 the amount of information sent by the task agents to the robot owner is $O(P \cdot T \cdot \log(N))$ bits. Step 3 is done in $O((P \log P) \cdot T)$ time. Finally, in Step 4, the price update is done in $O(P \cdot T)$ time.

5 Performance Analysis

The performance of the proposed decentralized multi-agent system model is compared with that of the optimization model for the production scheduling problem in the centralized scenario where the management of the plant is done by a single decision maker that controls all the information of the whole system.

5.1 Centralized Scheduling Model

The centralized production scheduling problem consists in minimizing the total cost of the tasks under the set of local task constraints and the constraint on the limited number of robots. Following analog reasoning used in modeling the local production scheduling problem addressed by each task in the decentralized scenario, we model the centralized problem as follows. The objective function to be minimized is the total cost of the tasks, that is, the social welfare $w = \sum_{i=1}^{P} z_i$, where z_i is the objective function (1) of problem P_i. The constraints to be fulfilled are the set of local constraints of each task i, that is, the set of constraints (2)–(6) of problem P_i, for each $i = 1, \ldots, P$, with the addition of the constraints on the limited number of robots, that is,

$$\sum_{i=1}^{P} n_i(k) \leq N, \quad k = 1, ..., T. \tag{13}$$

5.2 Simulation Results

The Multi-Agent System is simulated. The simulator and the negotiation algorithm are implemented in the C language using the CPLEX 8.0 callable library for the solution of problem P_i and are run on a PC with a 1.5 GHz dual-core duo CPU and 1 GB of RAM.

The production system is simulated with $P = 10$ tasks. The number of time periods of the planning horizon is $T = 100$, with the duration of each period $\Delta t = 1$. The external demand of task i is modeled as $d_i(k) = \bar{d}_i \cdot [1 + \delta d_i(k)]$, for $k = 1, \ldots, T$, where $\bar{d}_i$ is the average demand rate of the product produced by task i in the planning horizon and $\delta d_i(k)$ are values generated by the ARMA process with parameters $p_i = 2$ and $q_i = 2$ taken from the Example (2) of Gaalman (2006). In our numerical example, the average demand is $\bar{d}_i = 5$, for each product i.

We simulate the production system with different number of robots, N. In order to cover a wide range of cases (from very highly congested to very low-congested cases), we consider the values of N in the range $[\bar{N}(1-0.4), \bar{N}(1+0.4)]$, where $\bar{N}$ is the number of robots for which the average production rate of the system $\bar{N}\frac{1}{P \cdot T}\sum_{i=1}^{P}\sum_{k=1}^{T} r_i(k)$ (approximately) equals the total average demand rate $\sum_{i=1}^{P}\bar{d}_i$.

The values of the unitary production costs are fixed, noticing that if the unitary backlog cost $b_i(k)$ is not large enough in respect to the unitary manufacturing cost $c_i(k)$, and the robot cost $\rho_i(k)$, it may happen that, even if the system has enough capacity to clear all the demand, a positive backlog is found at the end of the planning horizon. For this reason, in the numerical case considered here, we assumed for each task i the following relation among the parameters of the cost function:

$$h_i(k) < c_i(k) + \frac{\rho_i(k)}{r_i(k)} < b_i(k) \tag{14}$$

The first inequality in (14) constrains the unitary holding cost $h_i(k)$ to be lower than the sum of the unitary manufacturing cost $c_i(k)$ and the robot cost per unit of allocated production capacity ($\frac{\rho_i(k)}{r_i(k)}$). The second inequality prohibits starvation as mentioned above. If these relations on the cost parameters hold, the backlog of the system at the end of the planning horizon can be used as a measurement of the congestion. For simplicity, we assume that the unitary costs of backlog, $b_i(k)$, inventory, $h_i(k)$, manufacturing, $c_i(k)$, and robot rental, $\rho_i(k)$, are constant and equal for all the tasks in all the time periods. In our case the costs are: $h_i(k) = 4$, $\rho_i(k) = 10$, $c_i(k) = 5$, $b_i(k) = 20$, $\forall i$, $\forall k$. Finally, the robot production capacity $r_i(k)$ is also assumed to be constant and equal for all the tasks and time periods. In particular, we assume $r_i(k) = 1$ for each task i and time period k. In this way, all the considered mixed integer problems are solved within a fraction of one second by CPLEX, and, hence, we are able to execute each simulation in less than two minutes; notice, however, that providing a CPU

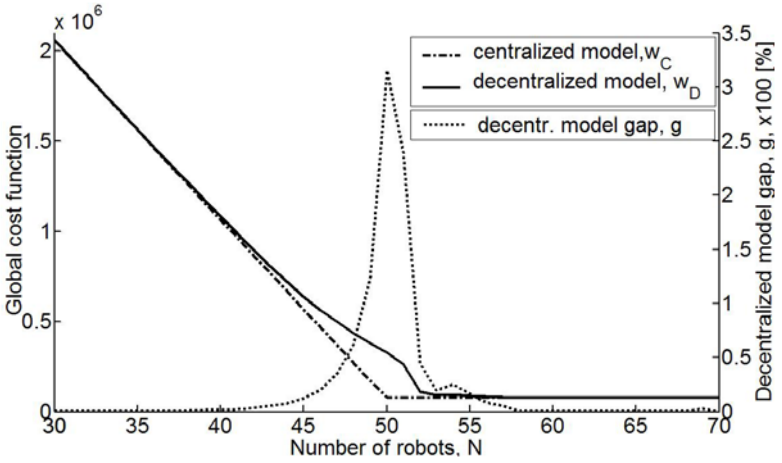


Fig. 2. Decentralized vs. centralized performance, number of tasks: $P = 10$

time efficient approach for the considered scheduling problem is out of the scope of this paper.

Let w_D be the best solution value of the total production cost (social welfare) obtained for the decentralized scenario by applying the proposed approach, and let w_C be the optimal solution value of the model for the centralized scenario. Performance of the decentralized model in respect to the centralized one is evaluated measuring the *gap* (in percentage) $g = [(w_D - w_C)/w_C] \cdot 100$, that provides an estimation of the relative extra-cost incurred in the decentralized scenario in respect to the centralized one for not having at disposal all the system information to optimally reconfigure and distribute the available robots.

Fig. 2, for $P = 10$, shows the trends of the social welfare w_D and w_C for the decentralized and centralized scenarios, respectively, and gap g, for increasing number N of robots in the range $[\bar{N}(1-0.4), \bar{N}(1+0.4)]$, with $\bar{N} = 50$ (since $\sum_{i=1}^{P} \bar{d}_i = 50$ and $\bar{N}\frac{1}{P \cdot T}\sum_{i=1}^{P}\sum_{k=1}^{T} r_i(k) = 50$). In particular, for $N \leq \bar{N}(1 - 0.15)$ or $N \geq \bar{N}(1 + 0.15)$, g is less than 5% and it decreases as N reaches the extremes of the considered range of values of N, for which g is less than 0.1%. In fact, when the total number of robots in the system is significantly lower than $\bar{N}$, the system is highly constrained, and both the decentralized and the centralized approaches are characterized by very high and comparable backlog costs. The total costs of the two approaches differ in practice in the values of the other cost components, that are indeed negligible in respect to the backlog cost. Oppositely, when the total number of robots in the system is significantly larger than $\bar{N}$, the local optimal solutions of the task agents' subproblems are likely to satisfy (resource) Constraints (7), and then the set of local solutions of the decentralized approach is equivalent to the centralized solution. Moreover, in this case, there is no backlog cost and then the total cost of both approaches is small. In the intermediate case, i.e., when N approaches $\bar{N}$, the gap g among the total costs of the decentralized and the centralized approach increases. This is due to the fact that the latter approach can take full advantage of having at disposal all the information, which allows it to satisfy the resource constraint with a smaller backlog cost in respect to that payed by the decentralized case.

6 Conclusions

A flexible manufacturing shop floor layout consisting of a set of mobile machines (robots) is considered in this paper. A multi-agent system coupled with an iterative auction based negotiation protocol is proposed to solve the problem of dynamic allocation of the robots to the tasks of the system in response to demand and/or cost fluctuations. For simplicity, the tasks are assumed independent, but the solution approach can also be applied to the case of tasks with precedence relations (see, e.g., Kutanoglu, 2006). The performance of the proposed decentralized solution is then compared with the same (optimal) obtained by using a centralized planner: the intuitive result that the two approaches provide a different cost only when the production capacity of the system is approximately equal to the average demand is noticed. This is the case in fact, when the conflict in the resource allocation is often violated and its proper management (possible only if the decision maker has all the system's information) makes the difference.

References

Barahona, F., Anbil, R.: The volume algorithm: producing primal solutions with a subgradient method. Math. Prog. 87, 385–399 (2000)

Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. Concurrency and Computation: Practice and Experience 14, 1507–1542 (2002)

Chen, H., Chu, C., Proth, J.M.: An improvement of the Lagrangean relaxation approach for job shop scheduling: A dynamic programming method. IEEE Trans. on Rob. and Autom. 14, 786–795 (1998)

Chevaleyre, Y., Endriss, U., Estivie, S., Maudet, N.: Multiagent resource allocation with k-additive utility functions. In: Proc. DIMACS-LAMSADE Workshop on Comp. Science and Decision Theory, Annales du LAMSADE, pp. 83–100 (2004)

Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S.: Issues in multiagent resource allocation. Informatica 30, 3–31 (2006)

Christensen, J.H.: Holonic manufacturing systems: Initial architecture and standards directions. In: Proc. 1st Euro Workshop on Holonic Manufacturing Systems, HMS Consortium, Hannover, Germany, December 1 (1994)

Clearwater, S.H.: Market-based control: a paradigm for distributed resource allocation. World Scientific Publishing Co., Inc, River Edge (1996)

Gaalman, G.: Bullwhip reduction for ARMA demand: The proportional order-up-to policy versus the full-state-feedback policy. Automatica 42, 1283–1290 (2006)

Held, M., Wolfe, P., Crowder, H.D.: Validation of subgradient optimization. Math. Prog. 6, 62–88 (1974)

Helms, E., Schraft, R.D., Haegele, M.: Robwork: Robot assistant in industrial environments. In: Proc. 11th IEEE workshop ROMAN 2002, pp. 399–404 (2002)

Huang, P.Y., Chen, C.S.: Flexible manufacturing systems: An overview and bibliography. Prod. and Inv. Mng. 27, 80–90 (1986)

Kraus, D.S.: Strategic negotiation in multiagent environments. MIT Press, Cambridge (2001)

Kurose, J.F., Simha, R.: A microeconomic approach to opt. resource allocation in distributed comp. systems. IEEE Transactions on Computers 38, 705–717 (1989)

Kutanoglu, E., Wu, S.D.: Incentive compatible, collaborative production scheduling with simple communication among distributed agents. Int. J. of Prod. Res. 44, 421–446 (2006)

Roundy, R.O., Maxwell, W.L., Herer, Y.T., Tayur, S.R., Getzler, A.W.: A price-directed approach to real-time scheduling of production operations. IIE Transactions 23, 149–160 (1991)

Schneider, J., Apfelbaum, D., Bagnell, D., Simmons, R.: Learning opportunity costs in multi-robot market based planners. In: Proc. 2005 IEEE International Conference on Robotics and Automation, pp. 1151–1156 (2005)

Selim, H.M., Askin, R.G., Vakharia, A.J.: Cell formation in group technology: review, evaluation and directions for future research. Comp. and Ind. Eng. 34, 3–20 (1998)

Sycara, K.P.: Multiagent Systems. AI Magazine 19(2), 79–92 (1998)

Vulkan, N., Jennings, N.R.: Efficient mechanisms for the supply of services in multi-agent environments. Decision Support Systems 28(1-2), 5–19 (2000)

A Study on Real-Virtual Interaction Method for Production Scheduling Using Model Plant

Yi Qian, Nobutada Fujii, and Toshiya Kaihara

Graduate School of Engineering, Kobe University
1-1 Rokkodai-cho, Nada, Kobe, Hyogo, 657-8501 Japan
senki@kaede.cs.kobe-u.ac.jp

Abstract. There is a great deal of external fluctuation such as order change or delayed delivery of materials, and uncertain components such as machine trouble or process delay in the real shop floor. It is difficult to execute real production according to the schedule which is established in advance. Therefore, a new production scheduling technique which can consider both external fluctuation and uncertain component existing in the real shop floor is required. In this paper, a real-virtual interaction method for production scheduling is proposed, which can adaptively and effectively consider both dynamic external fluctuation and uncertain component by using model plant in the real system and computer simulation in the virtual system to interact with each other. The effectiveness of real-virtual interaction method is verified by experiments on flow shop problem.

Keywords: model plant, real-virtual interaction method, production scheduling.

1 Introduction

To achieve effective operation of production systems, various production control systems have been proposed. It is necessary to adapt the production activity to suit market trends. Changes in market demands require more efficient control of production facilities. Thus, it is important to choose effective method for manufacture by taking the characteristics of the objective systems. Additionally, there is a great deal of external fluctuation such as order change or delayed delivery of materials, and uncertain components such as machine trouble or process delay in the real shop floor. It is difficult to execute real production according to the schedule established in advance. Therefore, a new production management technique which can adaptively and effectively consider both external fluctuation and uncertain component existing in the real shop floor is required.

Manufacturing system simulation has been utilized for evaluation and improvement of production system for a long period of time. However, simulation is usually executed inside of virtual system (computer system). Though computer simulation technology has been improved to achieve objectives in manufacturing systems, there are still many things existing in real system, which cannot be well described in the virtual system. In order to solute the gap between real system and virtual system, there are some experiments [1-4] which use the physical model (model plant) existing in real space instead of using only computer simulations. By the results of these experiments, lots of advantages are verified by using the model plant. Not only simulation progress can be observed in the physical system, but also physical unpredicted fluctuation such as machine trouble or process delay can be described in detail compared with the computer simulation proceeded only in virtual system. However, these studies using the model plant usually are limited to layout design for education [1] or verification for virtual system [2], few papers can be found which are applied to production management or production scheduling.

In this study, a real-virtual interaction based production management is proposed, which can consider both external fluctuation and uncertain component existing in the real shop floor. As the first step, in this paper, a real-virtual interaction method is applied to production scheduling, which uses model plant existing in real system and the computer simulation existing in virtual system to interact with each other. The effectiveness of the real-virtual interaction method is verified by experiments on flow shop problem.

2 Real-Virtual Interaction Manufacturing System

In this section, Real-Virtual Interaction Manufacturing System concept is explained, which aims to realize agility and adaptability to uncertain component existing in the real shop floor. First of all, 1. Non stop, 2. Flexibility online and 3. Maintenance online are strongly demanded for an intelligent system. As the artificial intelligence progressing of machine control system and IT communication technique, the components of manufacturing system can become more intelligent and handle information or make decision independently. Based on these engineering revolutions, there are lots of new manufacturing system concepts proposed such as Holonic Manufacturing System (HMS) [5] and Biological Manufacturing System (BMS) [6].

2.1 The Concept of Real-Virtual Interaction Manufacturing System

Proposed Real-Virtual Interaction Manufacturing System is based on these studies of HMS and BMS, aiming for adaptability and agility dealing with both dynamic external fluctuation and uncertain component existing in real shop floor. In the proposed system, decentralized autonomous agents are introduced, and they plan production scheduling in virtual system and instruct real production in real system simultaneously. In order to realize the above situation, agents have to own information both in the real system and in the virtual system. The agents use the information in real system for instructing real production and recording the actual production results which are fed back from the real shop floor. On the other hand, the agents use the information in virtual system for computer simulation when production scheduling is needed. Fig. 1 is showing the concept of Real-Virtual Interaction Manufacturing System. Fig. 1 (a) is showing equipments in real system connected to each agent, which are used for instructing real production and receiving actual results fed back from the real

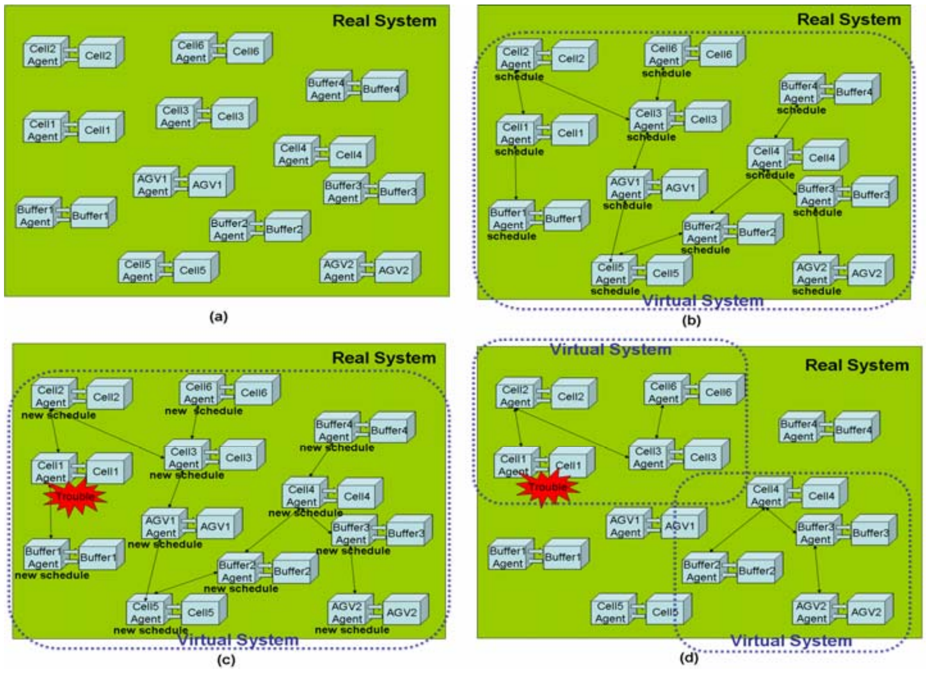


Fig. 1. The concept of Real-Virtual Interaction Manufacturing System

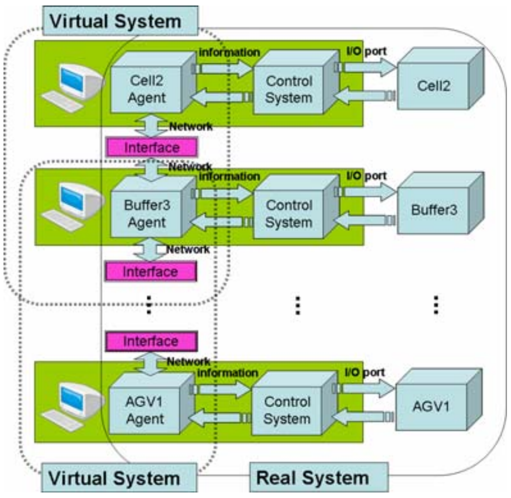


Fig. 2. Implementation of Real-Virtual Interaction Manufacturing System

shop floor. Fig. 1 (b) is showing that virtual system is constructed and schedule is made by the communication of agents when production scheduling is necessary. Fig. 1 (c) is showing that virtual system is constructed again when machine trouble occurs in the real system, and new schedule is established by the communication of agents using the present states in the real shop floor. Fig. 1 (d) is showing that it is not always necessary to consider the global optimum for whole manufacturing system, and new schedule can be planed faster by using a part of equipments which are influenced by the fluctuation.

In order to realize the Real-Virtual Interaction Manufacturing System, implementation method shown in Fig.2 is proposed. Each equipment in the real system is connected to each agent composed in an independent computer terminal, and information can be exchanged via network when virtual system is constructed for establishing schedule by agents. Furthermore, external fluctuation such as machine trouble or machine importing can be treated flexibly in decentralized manner.

2.2 Modeling of Agents

As we mentioned in section 2.1, agents are used to construct virtual system for schedule and monitor the real process of real system, so these agents need to have information both in the real and virtual systems, and information is always interacted between the real and virtual systems. As showed in Fig.3, agents use the information in the real system for instructing real production and recording the actual results which are fed back from the real shop floor. Information in real system is changing by the actual production results in real shop floor. On the other hand, agents use the information in the virtual system for computer simulation when production schedule is demanded. Information in the virtual system is also changing by the progress of computer simulation.

In addition, not only all equipments (machines, AGVs, etc.) are corresponding with each agent, but also order agents and work agents are exist to compose the system. These three types of basic agents are structured using object-oriented concepts like aggregation and specialization.

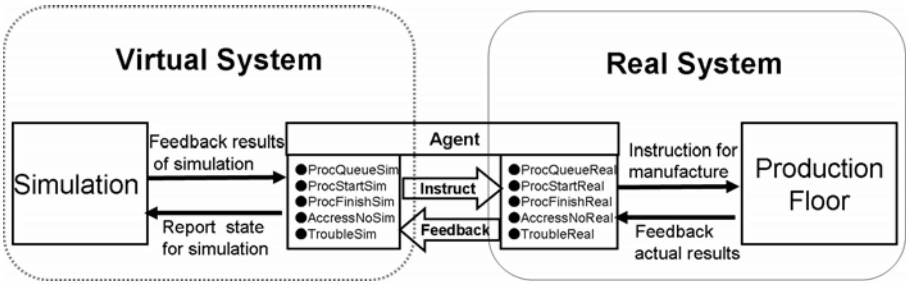


Fig. 3. Agent model

3 Model Plant

The left part of Fig.4 is showing the model plant used in this study, which is copyright by Fischertechnik [7]. It comprises four Milling Machines, one Conveyor, four Buffers and one Automated Storage/Retrieval System (AS/RS). There have motors,

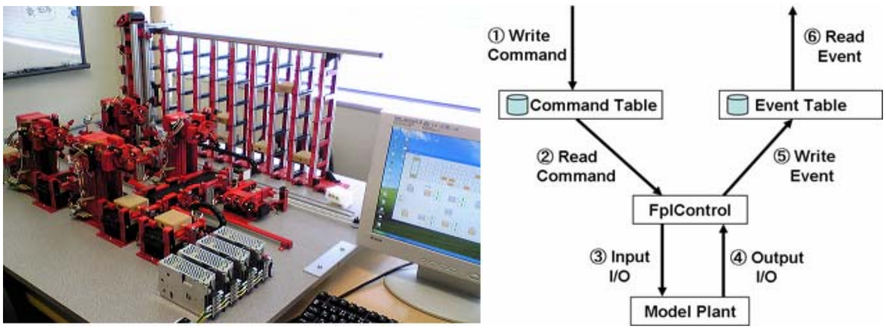


Fig. 4. Model plant and Control system

sensors, actuators, and control systems in all of these parts. The right part of Fig. 4 is showing the control system for model plant. The control process comprises the six steps (1-6) to operate model plant and receive the actual feedback results from model plant. FplControl is the control software of model plant and computer terminal by using I/O port, which is produced based on Microsoft Visual Basic [8]. Command Table is used for receiving commands input from computer terminal and sending the commands to the FplControl to operate model plant. Event Table is used for receiving events which are fed back automatically from the FplControl when commands are executed.

4 Real-Virtual Interaction Manufacturing System Using Model Plant

There are various unexpected fluctuations existing in the model plant using as the real shop floor to realize our proposition in the section 2. However, the control system of model plant can not be separated dispersedly at present, so the Real-Virtual Interaction Manufacturing System is realized as a centralized system as the first step. And for the global optimum of present centralized system, a GA (Genetic Algorithm) based scheduler is used for make schedule in the virtual system, which is connected to all agents. Fig.5 is showing the configuration of Real-Virtual Interaction Manufacturing System using model plant. All of the equipments in model plant are connecting with each agent by the control system. Information of real system is fed back to the virtual system when scheduling request occurs, and information of virtual system instructs real shop floor when a new schedule is established by computer simulation. In addiction, for the action as next step, schedule will be built and modified via the interaction of agents, which have each scheduler under a decentralized Real-Virtual Interaction Manufacturing System.

4.1 Virtual System Modification by Real-Virtual Interaction

Though the process of manufacture can be forecasted in computer simulations by using parameters measured in advance, time lag usually happens between prospects

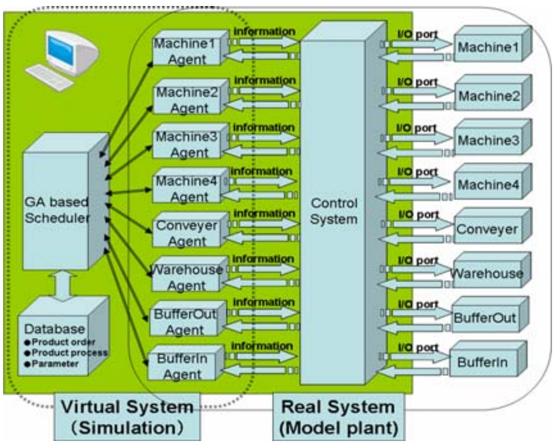


Fig. 5. Implementation of Real-Virtual Interaction Manufacturing System using Model Plant

and actual results because of various unexpected fluctuations existing in real shop floor. Thus, it is needed to modify these parameters frequently along with the variation in real system. Formula (1) and (2) are used for the virtual system parameter modification based on the time gap between prospect and actual result. Δt_m^p means the time lag of the same action such as transportation or work process. $t_{m,virtual}^p$ is the prospect of an action at time *m*. $t_{m,real}^p$ means the actual result of the action at time *m*. *k* is used to adjust the modifying speed of parameters.

$$\Delta t_m^p = t_{m,real}^p - t_{m-1,virtual}^p \tag{1}$$

$$t_{m,virtual}^p = t_{m-1,virtual}^p + k \times \Delta t_m^p \tag{2}$$

In addition, it also needs to change whole configuration of the virtual system when heavier fluctuation (such as machine trouble or machine recovery) happens in the real system, which is called as virtual system configuration modification. New schedule will be established automatically by the computer simulations based on GA when virtual system is reconstructed.

4.2 Real System Modification by Real-Virtual Interaction

Virtual system is changing by the feedback from real system, and computer simulation is executed automatically when a new schedule is required. The new schedule is saved in the virtual information of each agent when scheduling accomplished, and information for instructing real production will be modified automatically by the instruction from virtual information. Then, real production always proceeds according to the current schedule.

5 Experiments

In this section, experiments are proceeded to verify the feasibilities of the proposal. Flow shop problem is used to verify parameter modification in virtual system when there is time gap between prospect and actual result. Flexible flow shop problem is used to verify configuration modification in virtual system when there is machine trouble or machine recovery in real system.

5.1 Experiments in Flow Shop Problem

The experiments in a two-machines two-processes flow shop problem are executed. The experiment scenario is: works are picked up from warehouse by AS/RS and transported to Machine 1 by conveyor to start the first process, when the process 1 is finished, works are transported to Machine 2 by conveyor again to start the second process, then works are transported to warehouse by conveyor and picked off by AS/RS. 8 works (A001-A008) are introduced in the experiments, and GA is used to make schedule to minimize make-span in consideration of both the transportation of works and buffer limit of machines by the feedback information from agents.

The results of experiment are showed in Table 1 and Table 2. "Out, Tr1, Pr1, Tr2, Pr2, Tr3, In" are expressing "pickup by AS/RS, transporting to Machine 1, process 1 by Machine 1, transporting to Machine 2, process 2 by Machine 2, transporting to

Table 1. Comparison with S.D. between prospect and actual result (Sec.)

Pattern	*k*	*S*	Out	Tr1	Pr1	Tr2	Pr2	Tr3	In	Sum of time lag
1	1.0	5	1.0	1.7	0.2	2.6	0.2	2.8	1.0	52.2
2	0.6	5	1.4	1.7	0.3	0.7	0.3	1.0	0.7	43.5
3	0.2	5	1.2	0.7	0.3	3.4	0.3	3.0	0.6	51.5
4	1.0	10	1.4	1.5	0.3	2.2	0.3	4.0	4.6	63.0
5	0.6	10	1.9	0.7	0.3	0.9	0.3	2.1	1.0	48.0
6	0.2	10	1.5	0.7	0.3	4.5	0.3	3.3	0.6	56.5

Table 2. Change of make-span by prospect and actual result (Sec.)

Pattern	Actual result	Change of make-span by prospect									
		Time	0	146	212	357	352	735	838	976	1086
1	1224.7	Make-span	1148	1210	1216	1180	1190	1190	1201	1203	1214
		Time	0	110	213	358	605	739	917	1025	1150
2	1222.5	Make-span	1148	1187	1172	1198	1192	1195	1207	1212	1221
		Time	0	109	193	357	499	630	806	979	1056
3	1224.3	Make-span	1148	1155	1176	1183	1191	1195	1200	1206	1215
		Time	0	194	527	783	921	1057			
4	1227.4	Make-span	1148	1200	1190	1197	1211	1220			
		Time	0	193	627	912	1092				
5	1216.4	Make-span	1148	1183	1193	1200	1215				
		Time	0	193	585	866	1080				
6	1228.0	Make-span	1148	1175	1175	1201	1215				

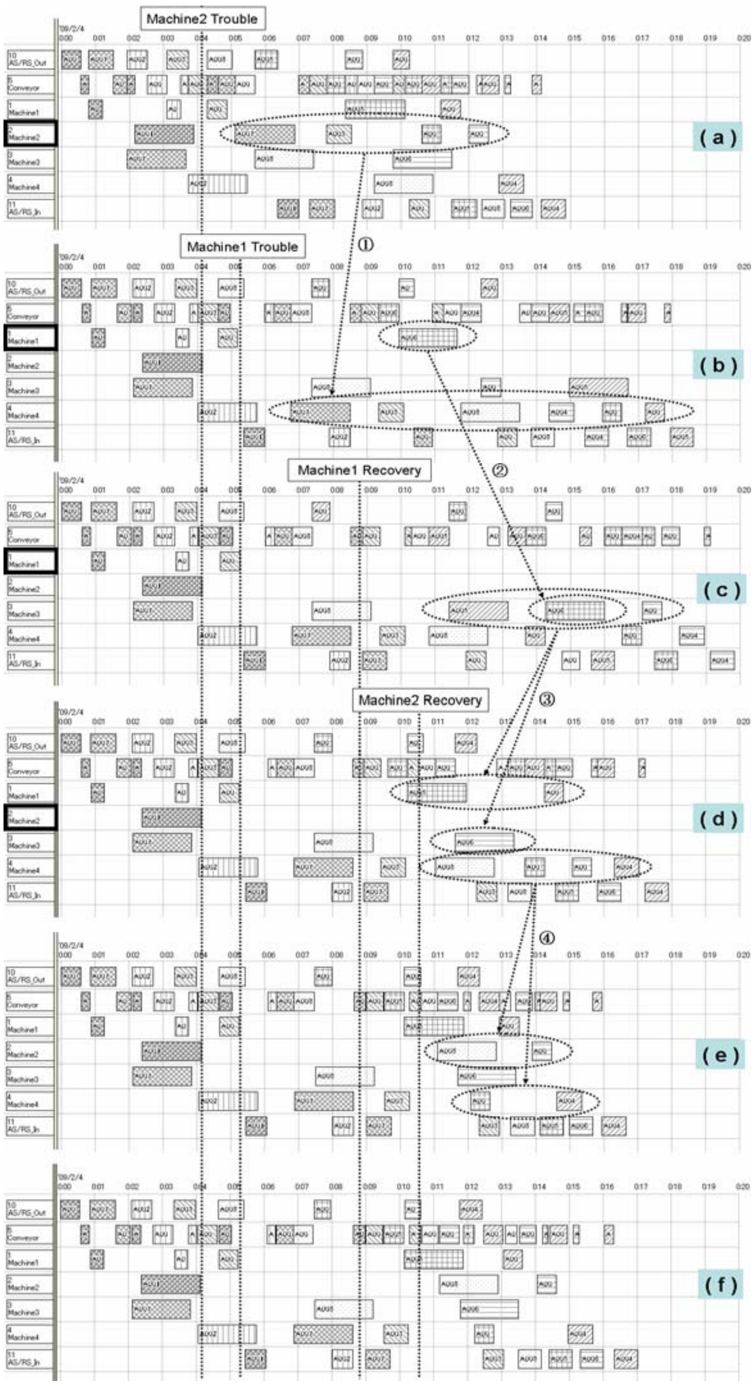


Fig. 6. Change of schedule and actual result with Machine 2, Machine 1 trouble and Machine 1, Machine 2 recovery

warehouse, pickoff by AS/RS" for works respectively. 6 patterns are executed in the experiments, *k* is used to adjust the modifying speed of parameters, and *S* is the standard of total time lag (make-span is modified using modified parameters by computer simulation when total time gap exceeded *S*). From Table 1, it is verified that time lag between prospect and actual result is contained by parameter modification for all patterns, and pattern 2 has the best result for controlling time lag. From Table 2, it is verified that prospect of make-span is renewed automatically when total time lag is beyond *S*, and make-span is approaching to the actual result gradually as parameters' modification. The experiment results show the effectiveness of parameter modification in the proposed method.

5.2 Experiments in Flexible Flow Shop Problem

Experiments in four-machines two-processes flexible flow shop problem are executed to verify the effectiveness of the proposed method when machine trouble or machine recovery happens in real system. The scenario and setting of works are the same as the flow shop problem, in addition, Machine 1 and Machine 3 for process 1, Machine 2 and Machine 4 for process 2 are introduced. The results of experiments in section 5.1 (k=0.6, S=5) are used to modify parameters of the virtual system for rescheduling when machine trouble or machine recovery is happened. The results of the experiments are showed in Fig. 6. Fig. 6 (a) depicts the initial schedule established in the virtual system. Fig. 6 (b) and (c) are showing the new schedule which is established automatically when Machine 2 or Machine 1 trouble happens. Fig. 6 (d) and (e) show the new schedule which are produced automatically when Machine 1 or Machine 2 is recovered. Fig. 6 (f) illustrates the actual result in the real system. By comparing with Fig. 6 (a) and (b), and Fig. 6 (b) and (c), it is observed that Machine 4 and Machine 3 can alternate Machine 2 and Machine 1 to process the remained works by the new schedule when there is machine trouble in the real system (dotted line 1 and 2). Compared with Fig. 6 (c) and (d), and Fig. 6 (d) and (e), it is revealed that Machine 1 and Machine 3, Machine 2 and Machine 4 can share the remained works by the new schedule when the machine is recovered (dotted line 3 and 4). In addition, compared Fig. 6 (f) with others, it is verified that real production in the real system proceeds according to the newest schedule established by real-virtual interaction. These results reveal the effectiveness of the proposed method to deal with machine trouble and machine recovery adaptively.

6 Conclusion

In this paper, we propose a Real-Virtual Interaction Manufacturing System which can deal with both dynamic external fluctuation and uncertain component existing in real shop floor. As the first step, a real-virtual interaction method for production scheduling is proposed, and it is verified the effectiveness of the proposal by experiments using flow shop and flexible flow shop problems. It is observed that parameter modification and configuration modification by real-virtual interaction are effective to deal with time gap and machine trouble or machine recovery.

In future, to pursuit more flexibility of Real-Virtual Interaction Manufacturing System, decentralized system will be constructed by distributing the control system of model plant. And schedule will be built and modified via the interaction of agents, which have each scheduler under the decentralized Real-Virtual Interaction Manufacturing System. The final goal of this study is to establish a new production management method which can adaptively and effectively deal with any external fluctuation and uncertain component existing in the real shop floor autonomously by the interaction of real system and virtual system.

References

1. Fang, X.D., Shivathaya, S.S.: Introducing Fundamental Concepts of Manufacturing Systems to Fresh Engineering Students by Physical Simulation of Automated Factory. Int. J. Engng. Ed. 13(2), 117–122 (1997)
2. Agre, J., Clare, L., Lee, J., Brandin, B., Hoskins, J., Perrone, M.: Autoconfigurable Distributed Control Systems. In: Proceedings of the Second International Symposium on Autonomous Decentralized Systems(ISADS 1995) (1995)
3. Leitao, P., Restivo, F.: Experimental Validation of ADACOR Holonic Control System. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS (LNAI), vol. 3593, pp. 121–132. Springer, Heidelberg (2005)
4. Ramamoorthy, K.: National Instruments Programmable Automation Controller for Reconfigurable Logic Control: Implemantation and Evaluation. Engineering Research Center for Reconfigurable Manufacturing Systems ERC/RMS University of Michigan (September 2007)
5. HMS consortium. Holonic Manufacturing System: Revolutionary of manufacturing with human, machine, system's flexible cooperation. Plant maintenance association of Japan (2004)
6. Okino, N.: Biological Manufacturing System: From concentration to decentralization. Asakura bookstore (1993)
7. Fischertechnik, `http://www.fischertechnik.com`
8. Microsoft Visual Basic, Express Edition (2005), `http://www.microsoft.com/japan/msdn/vstudio/express/`

Using an Agent-Supported Simulation Environment for Intelligent Manufacturing Systems

Nancy Ruiz, Adriana Giret, and Vicente Botti

Department of Information Systems and Computing, Polytechnic University of Valencia, Spain 46022
{nruiz,agiret,vbotti}@dsic.upv.es

Abstract. The manufacturing field is an area where the application of simulation is an essential tool for validating methods and architectures before applying them on the factory floor. Multiagent System technology has demonstrated its utility in manufacturing system modeling and implementation. Agenthood features such as proactivity, reactivity, and sociability may also be useful for associating them with the specific simulation needs of the new manufacturing requirements. In this paper, we present an Agent-supported Simulation Tool (tool uses both events and discrete time to control agent tasks) for Intelligent Manufacturing Systems applied to a real manufacturing enterprise case study. The main goal is to provide a flexible simulation tool that can be adapted to solve the new manufacturing requirements that appear in a real environment allowing the experts of manufacturing domains to optimize the resource usage and to have enough data to make decisions.

Keywords: Intelligent Simulation, Multiagent-Holonic Simulation, Intelligent Manufacturing Systems Simulation.

1 Introduction

Multiagent System technology has demonstrated its utility in manufacturing system modeling (CIIMPLEX [1], HOLOS Architecture[2]). Based on the features of agents (proactivity, reactivity, and sociability), it has been possible to apply this technology to Enterprise Integration and Supply Chain management. Since manufacturing requirements (specification) change according to customer needs, the simulation that has been widely applied to design manufacturing systems must also adapt to those needs [3]. Our proposal (SimIShopF) focuses on the improvement of the modeling and simulation processes of a Shop Floor, solving problems related to flexibility, control and knowledge distribution, complex behavior simulation, automatic creation and/or elimination of elements, and process execution according to the current system state. The main goal is to provide a flexible simulation tool that can be adapted to solve the new manufacturing requirements that appear in a real environment. This paper presents the use in a real case study of SimIShopF, which is a prototype used to validate an agent-supported architecture [4] for the simulation of Intelligent Manufacturing Systems [5]. In the second section, the global description of the simulation process are presented. In the third section, the results of using the simulation tool in the Study Case is presented. Finally, we present our conclusions.

2 SimIShopF: The Simulation Tool

SimIShopF is a tool that provides easy-to-use interfaces of an agent-suppported simulation architecture [4] for intelligent manufacturing systems [5] presented in previous works. SimIShopF was designed using the ANEMONA methodology [6]. It is based on the set of steps in a simulation study and the principles of Simulation Modeling proposed by Banks et al. [7]. SimIShopF provides interfaces for the Simulation Process, which is divided into two phases (Fig.1): *I) Model Creation, and II) Model Simulation.* In Phase I, a model is created by the User. This model is then used as input to Phase II, where the model is simulated by animating the behavior of its elements. When model simulation finishes, SimIShopF allows the User to apply metrics to evaluate the results.

The technology used to implement the SimIShopF includes: a) JAVA (Ver. 6.x), a standard programming language, b) JADE (Ver. 3.4), an agent development framework based on JAVA, which provides basic schemas to define agents and communication protocols, c) ECLIPSE (Ver. 3.2.2), a development framework that manages JADE, d) PROTEGÉ-OWL (Ver. 4.x), an ontology editor, and e) MySQL (Ver. 5.0), which provides a multi-threaded, multiuser and robust SQL database server (Structured Query Language).

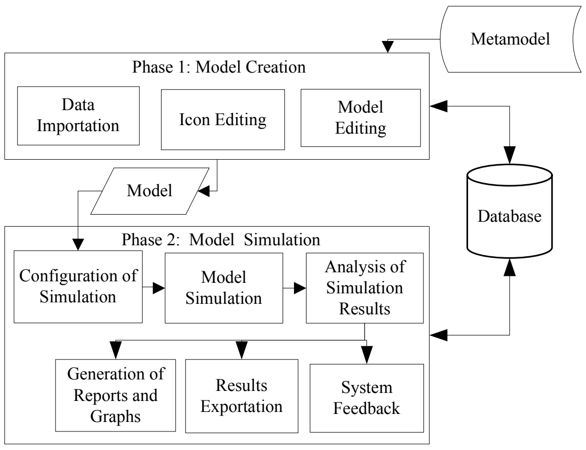


Fig. 1. Phases of Global Simulation Process

3 Using SimIShopF

In this paper it is used a Case Study from a Metal-mechanic Industry that manufactures product containers. The simulated experiments are focused on the validation of processes and resource usage in a new shop floor based on a current shop floor. Thus, the goal is to detect improvements to resource usage and design experiments to extend the shop floor capability and respond to unexpected or specific situations (for example, broken machines or machine acquisition, workers get sick or hired workers). This section presents the detailed activities that the Agents execute during the Model Creation (Phase I) and Model Simulation (Phase II) for the metal-mechanic industry Case Study. The model created in *Model Creation, Phase I*, will be simulated in *Model Simulation, Phase II.* This section also presents the main features of the modeling supported by agent interactions in SimIShopF (Fig.1) such as: a) how the metamodel is used during model creation, b) the interaction between the metamodel and the Agent-supported Simulator during the Simulation Process, c) the criteria to help the User to create a model, d) how the work centers are defined in a model, and e) how specific resource instances are linked to the model icons and the rules related to the organization modeling.

In Phase I [4], the User imports data from the real Shop Floor that are analyzed and included as attributes of predetermined icons (to represent workers, machines, tools and staff). Thus, The User creates a Model using that icons. Moreover, it is also defined the product container detail related to its production route and bill of materials. The route production includes the sequence definition (transition) between tasks. The Model definition (based on a metamodel) takes into account two levels: a) at upper-level, the physical installation resources (workers, machines, tools and staff) and groups are described, and b) at lower level, the capabilities, time restrictions, costs, capability grade, work center linked to resources are defined.

In this Case Study, based on the definition levels presented above, at upper level, the User identifies work centers and their resources and attributes. Thus, there were identified nine work centers that are grouped according to its main purpose. Table 1 shows their main purposes and resources. SimIShopF helps the User to create the Model by offering basic schemas for the definition of the nine work centers and their resources. When the Model is free of mistakes, it is ready to be used in *Phase II.* The Manufacturing System Model designed in Phase I, is internally an Agent Model that will be used to construct a new

Table 1. Main purpose of Work Centers and their Resources

Work Center	Purpose	Resources
H01, H21, H24	Cutting	5 Workers, 6 Machines, 1 Staff
S24, S26, S27	Welding	5 Workers, 4 Machines, 1 Tool, 1 Staff
P14	Painting	3 Workers, 2 Machines, 3 Tools, 1 Staff
P19	Packaging	4 Workers, 2 Machines, 1 Staff
P17	Washing	4 Workers, 2 Machines, 2 Tools, 1 Staff

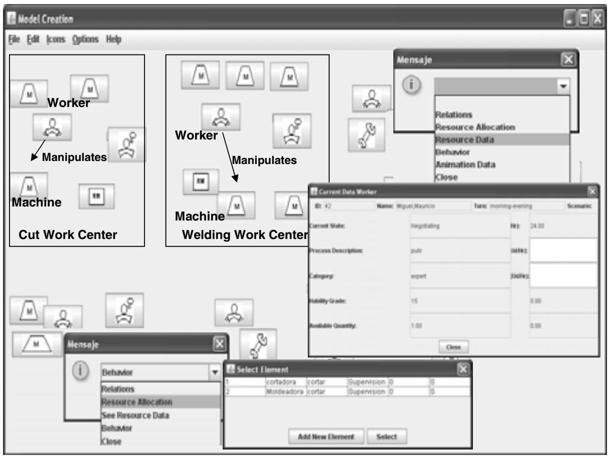
Multiagent System that interacts with the Simulation Tool in Phase II (Fig. 3). Thus, the User defines three experiments to simulate and validate the model. During simulation SimIShopF allows the User to: a) define the simulation speed, animation type, and stop condition, b) launch production orders, c) stop and pause the activity of specific resources and stop simulation process at any time, d) monitor the state of resources and production orders, and e) include "what-if" events (e.g. add resources, modify resource attributes).

In next section the obtained model and its simulation by using SimIShopF in this Case Study will be described. The results analysis of the model simulation also takes into account the main goals of manufacturing systems (reduce time and cost) and the evaluation of the resource type of the model elements.

3.1 Creating a Model

During Model creation, there is continuous interaction between the User and the Simulation Tool by means of interfaces that are easy- to-use and understand. Based on the problem description and the goals of the Case Study, the User creates the Model using the distribution of resources (workers, machines, staff) in the nine work centers presented in Table 1 by using icons. Thus, the User links instances to icons for defining the attributes to each resource such as name, capability grade (beginner, medium, expert), capacity per hour, cost per hour, etc. SimIShopF helps the User (Model Designer) during the design of this nine work centers and their resources by: a) Providing the User with different help options for adding elements during the model design. b) Defining an appropriate level of detail which allows the User to include basic information (resource attributes) that could help to easily describe real issues in the Manufacturing System that is represented. c) A functional design of the Database to efficiently store and recover the information imported from the real shop floor. d) Interfaces were the User can easily understand the kind of information required and the results obtained. During Phase I, the event detection method is applied to detect the state changes in the system [8].

During the Creation/Edition of a Model, the Modeler Agent offers the User basic schemas in accordance with the work center type (i.e., cutting, welding, finishing, moulding, thermic, chemical). Thus, the schema includes the type and minimum quantity of elements suggested (i.e., workers, machines, tools) and also the human resources who supervise the activities (Staff) for the nine work centers. In addition, the relations between elements (resources and staff) can be defined (manipulation, production, use, transformation, supervision). According to Table 1, Figure 2 shows the resources and relations of the cutting work center (H01) and the welding work center (S24) of the Model. Thus, the model defines the capabilities of the shop floor and fixed relations between resources in a work center. The dependencies between work centers are defined according to the product route. In comparison with current Simulation Tools, SimIShopF allows the User to design a customized Model that can allocate a specific resource instance to model icons and they can be grouped into work centers. The Modeler Agent offers instances of the resources (defined into the real shop floor) that can



Current Data Worker

ID: 42 Name: Miguel,Mauricio Turn: morning-evening Scenario:

Current State:	Negotiating	Hr):	24.00
Process Description:	pulir	(d/Hr):	
Category:	expert	(Ud/Hr):	
Hability Grade:	15		0.00
Available Quantity:	1.00		0.00

Close

Select Element

1	cortadora	cortar	Supervision	0	S
2	Moldeadora	cortar	Supervision	0	S

Add New Element Select

Fig. 2. Definition of Work Centers and their elements

execute a given activity in the work center (i.e., a cutter machine can made the cut activity). The work centers are dynamic groups, they also can be modified during model simulation. Before a Model can be simulated, it is verified by the Modeler Agent that informs the User when it is ready to be simulated. The verification is based on the Modeling Organization Theory [10].

The basic rules used to verify a work center definition include: a) **Behavior norms**. If a machine or tool must be manipulated by a worker, a worker must manipulate the machine or tool. Otherwise, if a machine or tool works automatically, no worker can manipulate that machine or tool. When a relation is defined, the rules include: i) if relation type is "manipulate", a worker only can manipulate machines or tools, and ii) if relation type is "supervise", a staff supervise workers, machines and tools. b) **Capability norms**. A worker instance and staff instance can be only linked to an icon in the model, a machine or tool instance can be linked to icons depending on the instance number that are defined in the database, and all icons must be linked to an instance to delimit their capacities. These rules ensure that the model has enough information for its simulation (Phase II). If some are not fulfilled, the model cannot be simulated.

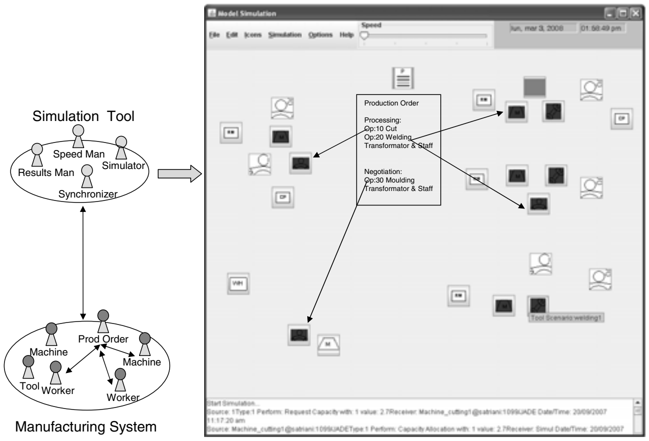
At the end of this phase, the Shop Floor Model includes the physical installation resources (workers, machines, tools and staff) presented above grouped into nine work centers (Table 1) and the capabilities, time restrictions, costs, capability grade of the resources. In this way, the modeled resources are traduced as individual agents in Model Simulation.

3.2 Model Simulation

This section presents the detailed activities that the Agents execute during the Model Simulation (Phase II). The simulation process is based on the agent-supported architecture presented in [4].

The Manufacturing System Model designed in Phase I, is internally an Agent Model that will be used to construct a new Multiagent System that interacts with the Simulation Tool (another Multiagent System) in Phase II (Fig. 3). The Synchronization Manager Agent supervises the model simulation according to the current state of the System and the Model generated as a result of the first phase. The Simulator Agent controls the activity of an agent set (that is in charge of the simulation process). The Simulator Agent allows the User to configure the simulation (definition of the simulation speed, animation type, and stop condition) before simulation starts. The selected speed affects the global clock that is used by all the agents.

For this Case Study, the User (focused on the reduction of time and cost) measures the nine work centers by launching production orders according to three experiments. The three experiments guide the validation of results for the Case Study. When the User launches a production order, the product, required quantity, required date, product route and its own shop floor configuration that will be used to produce that order are defined (Fig. 4). This feature provides more flexibility in order to design experiments and validate the User hypothesis. In addition, the configuration can be defined in two ways: a) the User can select specific resources linked to each one of the route operations or b) the User allows the order agent to negotiate about scheduling with the available resources. For the last option a negotiation technique based on auctions has been implemented. The goal of the negotiation among Orders and Resources is to optimize Resource usage during a Task according to the Order restrictions (capacity vs time and quality). The three experiments were defined by using different types of techniques for task assignment in five orders: i) direct assignment of tasks to specific resources, ii) mix direct assignment and assignment of task by negotiation, and iii) assignment of task by negotiation.



Start Simulation...
Source: 1Type:1 Perform: Request Capacity with: 1 value: 2.7Receiver: Machine_cutting1@satriani:1099/JADE Date/Time: 20/09/2007 11:17:20 am
Source: Machine_cutting1@satriani:1099/JADEType:1 Perform: Capacity Allocation with: 1 value: 2.7Receiver: Simul Date/Time: 20/09/2007

Fig. 3. Agent-supported Model Simulation of an Intelligent Shop Floor

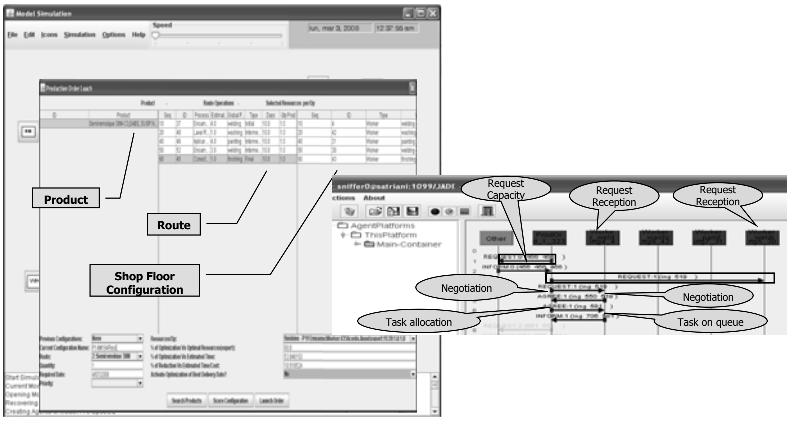


Fig. 4. Launching Production Orders and Internal Agent Interaction

In addition, when the User launches a production order, the Manufacturing System executes its own tasks (Fig. 4): production orders and resources keep in touch for task assignment, the resources inform the production orders about the progress of the accepted tasks. A Work Center requires inputs (raw material/semiproducts) that are processed to produce outputs (semiproducts). Therefore, both Raw Material and SemiProduct elements are included like icons in the simulation process according to the progress of production orders. The Simulator Agent receives the state changes of the Manufacturing System (i.e. machine X accepts a task, worker Y is stopped, etc.) and they are translated into events. The Simulator Agent informs the Animation Manager Agent of the event, who animates the icons linked to that event. The Input/Output Data Analyzer Agent analyzes the results when the simulation has finished. The Icon Manager Agent reuses the analyzed information, which is linked and included in the icons. Moreover, during simulation it is possible to include "what-if" events like add a new element or change the capability per hour of a resource.

During Simulation it is possible for the User to trace the current behavior of each one of the model elements. Thus, it is possible to observe: a) when a resource (staff, worker, machine or tool) is working, negotiating or has stopped, b) the current operation and state of a semiproduct (semiproduct indicates the current state of a product), c) the current animation configuration of any element, and d) the production progress of an order.

3.3 Evaluation of Results

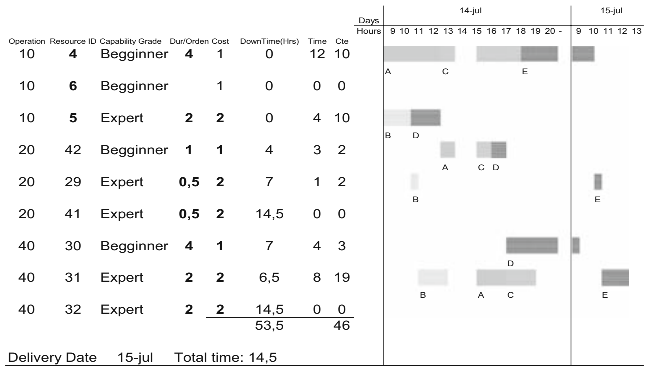
The User measures and evaluates the results of the simulation of the three experiments based on manufacturing goals: reduce time and cost. In addition it has been included, the evaluation according to the instance type of the model resources (beginner, medium and expert). Thus, it is possible to evaluate models, shop floor configurations and production data. The analysis allows the User to appreciate capability grade of workers and equipment (machinery and tools) used.

Model Evaluation. The model evaluation can be made before and after a model is simulated. A model is evaluated based on the instance capability grade (beginner, medium or expert) of the model elements. In an ideal model all resources are "experts". For this proposal it is assumed that a resource with a capability grade *expert* can carry out tasks with a higher quality grade and requires less time than a resource with a capability grade *beginner* or *medium*.

Evaluation of Shop Floor Configurations. During Model simulation, when a User launches a production order, the User can specify the shop floor configuration to be used according to its needs. Thus, it is possible to evaluate previous configurations linked to the product. A shop floor configuration can be evaluated before and after it was used for a production order. Thus, it is possible to observe three global scenarios: a) Manual Task Assignment using the same configuration, b) Assignment based on negotiation techniques, and c) Combination of assignment techniques.

Analysis of Production Data. The analysis equations of a Flexible Manufacturing System [11] have been used to identify indicators to define equations to evaluate and analyze results. Thus, the simulation results are used to measure the resources and model performance both historical and current. The Results Agent provides reports to show the global or individual behavior of the model resources. Some of the results are: a) production order cost, b) production time of an order, c) total down time of the shop floor, and d) used capacity of the model resources.

Figures 5 to 7 show the results of the simulation of the three experiments presented above. Figure 5 presents the results when direct assignment of tasks technique is used. In this assignment type, when the User selects resources, this



Operation	Resource ID	Capability Grade	Dur/Orden	Cost	DownTime(Hrs)	Time	Cte
10	**4**	Begginner	**4**	1	0	12	10
10	**6**	Begginner		1	0	0	0
10	**5**	Expert	**2**	**2**	0	4	10
20	42	Begginner	**1**	**1**	4	3	2
20	29	Expert	**0,5**	**2**	7	1	2
20	41	Expert	**0,5**	**2**	14,5	0	0
40	30	Begginner	**4**	**1**	7	4	3
40	31	Expert	**2**	**2**	6,5	8	19
40	32	Expert	**2**	**2**	14,5	0	0
					53,5		46

Delivery Date 15-jul Total time: 14,5

Fig. 5. Use of Resources using direct task allocation

Operation	Resource ID	Capability Grade	Time	Cost/hr	DownTime(Hrs)	Ttime	Cost
10	**4**	Begginner	**4**	0,8	0	8	6,4
10	**6**	Begginner		0,8	0	4	3,2
10	**5**	Expert	**2**	**2,4**	0	4	9,6
20	42	Begginner	**1**	**0,8**	4	1	0,8
20	29	Expert	**0,5**	**2,4**	7	1,5	3,6
20	41	Expert	**0,5**	**2,4**	4	0,5	1,2
40	30	Begginner	**4**	**0,8**	10,5	0	0
40	31	Expert	**2**	**2,4**	3,5	6	14
40	32	Expert	**2**	**2,4**	4,5	4	9,6
					33,5		49

Delivery Date: 14-jul Total time:10,5

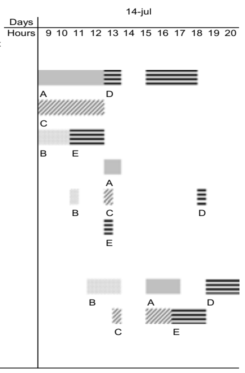


Fig. 6. Use of Resources using a mix of direct task allocation and negotiation

Operation	Resource ID	Capability Grade	Time	Cost	DownTime(Hrs)	Ttime	Cte
10	**4**	Begginner	**4**	0,8	0	4	3,2
10	**6**	Begginner		0,8	0	4	3,2
10	**5**	Expert	**2**	**2,4**	0	6	14
20	42	Begginner	**1**	**0,8**	4	1	0,8
20	29	Expert	**0,5**	**2,4**	5	1,5	3,6
20	41	Expert	**0,5**	**2,4**	4	0,5	1,2
40	30	Begginner	**4**	**0,8**	5	2	1,6
40	31	Expert	**2**	**2,4**	2,5	6	14
40	32	Expert	**2**	**2,4**	4,5	2	4,8
					25		47

Delivery Date: 14-jul Total time:8,5

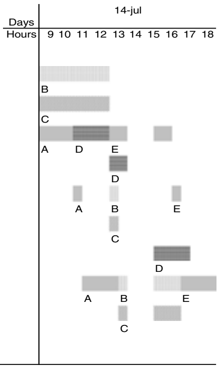


Fig. 7. Use of Resources based on task negotiation

activity requires little time to assign the tasks of a product route. However, this situation gives rise to some problems like bottle necks, resource sub-explotation and the increase of down time on the shop floor. To solve these problems the User constantly verifies the system. This technique misuses the potential of the agent paradigm and the negotiation protocols.

Figure 6 presents the results when direct assignment of tasks and assignment based on negotiation are used. In this combination down time reduction and resource usage are possible. The manual assignment fosters task delays and bottle necks. The negotiation techniques used for task assignment promote the work distribution. However, the prolonged use of manual assignment increases the risk of sub-optimization of the available capacity and requires more time for monitoring the model simulation and make decisions.

Figure 7 presents the results when task assignment based on negotiation is used. In this assignment type the bottle necks are reduced, resource usage is improved and down time is reduced. However, this technique requires more time to assign tasks. The use of appropriate algorithms helps to make decisions during task assignment in a short time. Moreover, this technique reduces bottle necks and improves the task distribution among available resources. It also reduces the monitoring time of the User.In this way, the User identify that the best option for optimize the resource usage is the use of task assignment based on negotiation.

4 Conclusions

In this paper the application of simulation tool has been presented, which allows the validation of the agent-supported simulation architecture for the creation and simulation of agent-supported models for intelligent manufacturing systems presented in previous works [4,5,12]. Both agent-supported architecture for simulation and an intelligent manufacturing metamodel have been proposed to solve the requirements of the new manufacturing era. The tool provides enough flexibility to design complex models and experiments. SimIShopF includes the main tasks of a simulation tool like: model creation, model simulation, animation and distribution of elements. Moreover, the basis for evaluating models and shop floor configurations have been presented. In addition, we have shown the scenarios and advantages of using the agent paradigm to simulate the complex behaviors of an Intelligent Manufacturing System. One of the main contributions of this work, is to provide a tool that allows the User to take advantage of the agent paradigm by means of easy-to-use interfaces. Since the agent-supported simulation of manufacturing systems is a wide-ranging field, some future work includes: creating a recommendation system of configurations, simulating warehouse behavior, adding a new module to link the database of the simulation tool to the database of an ERP, among others.

Acknowledgments. The authors would like to thank the financial support received from the Spanish government and FEDER funds under CICYT TIN2006-14630-C03-01 and TIN2005-03395 projects is gratefully acknowledged. This project is also partially supported by the Mexican CONACYT under research grant N178874, and by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

1. Peng, Y., et al.: A multi-agent system for enterprise integration. In: Proc. of 3rd Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agents, London, pp. 155–169 (1998)
2. Rabelo, R., Camarinha-Matos, L.: Multi-agent based dynamic scheduling. Int. J. on Robotics and Computer Integrated Manufacturing II(4), 303–310 (1994)
3. Shen, W., Norrie, D., Barthes, J.: Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. Taylor and Francis, London (2001)
4. Ruiz, N., Giret, A., Botti, V.: An Agent-Supported Simulation Architecture for Manufacturing Systems. In: Agent-Directed Simulation, Part of SpringSim 2007, pp. 63–70. ACM Press, Northfolk (2007)

5. Ruiz, N., Giret, A., Botti, V.: Agent-Supported Modeling and Simulation for Manufacturing Systems. In: IInd Workshop on Industrial Applications of Distributed Intelligent Systems, pp. 63–76. CAEPIA, Salamanca (2007)
6. Botti, V., Giret, A.: ANEMONA. A Multi-Agent Methodology for Holonic Manufacturing Systems. Springer Series in Advanced Manufacturing, vol. XVI. Springer, New York (2008)
7. Banks, J.: Handbook of Simulation:Principles, Methodology, Advances, Applications, and Practice. John Wiley and Sons, New York (1998)
8. Carrascosa, C., Terrasa, A., García-Fornes, A., Espinosa, A., Botti, V.: A metareasoning model for hard real-time agents. In: Marín, R., Onaindía, E., Bugarín, A., Santos, J. (eds.) CAEPIA 2005. LNCS, vol. 4177, pp. 42–51. Springer, Heidelberg (2006)
9. Ruiz, N., Giret, A., Botti, V.: Holonic Architecture for a Multiagent-Based Simulation Tool. In: 9th Int. Conf. on Enterprise Information Systems, pp. 395–398. ACM Press, Madeira (2007)
10. Wagner, J., Hollenbeck, J.: Comportamiento Organizativo. Thomson-Paraninfo, Spain (2004)
11. Stam, A., Kuula, M.: Selecting a flexible manufacturing system using multiple criteria analysis. Int. J. of Production Research 29(4), 803–820 (1991)
12. Ruiz, N., Giret, A., Botti, V.: Towards an Agent-based Simulation Tool for Manufacturing Systems. In: 11th IEEE Int. Conf. on Emerging Technologies and Factory Automation, pp. 797–804. IEEE Press, Prague (2006)

A Study on Real-Time Scheduling for Holonic Manufacturing Systems – Determination of Utility Values Based on Multi-agent Reinforcement Learning

Koji Iwamura, Norihisa Mayumi, Yoshitaka Tanimizu, and Nobuhiro Sugimura

Osaka Prfecture University, Gradutae School of Engineering, 1-1 Gakuen-cho, Nakaku, Sakai, Osaka, 599-8531, Japan
{iwamura,tanimizu,sugimura}@me.osakafu-u.ac.jp

Abstract. This paper deals with a real-time scheduling method for holonic manufacturing systems (HMS). In the previous paper, a real-time scheduling method based on utility values has been proposed and applied to the HMS. In the proposed method, all the job holons and the resource holons firstly evaluate the utility values for the cases where the holon selects the individual candidate holons for the next machining operations. The coordination holon secondly determine a suitable combination of the resource holons and the job holons which carry out the next machining operations, based on the utility values. Multi-agent reinforcement learning is newly proposed and implemented to the job holons and the resource holons, in order to improve their capabilities for evaluating the utility values of the candidate holons. The individual job holons and resource holons evaluate the suitable utility values according to the status of the HMS, by applying the proposed learning method.

Keywords: Holonic Manufacturing Systems, Real-time Scheduling, Multi-agent Reinforcement Learning, Coordination.

1 Introduction

Recently, automation of manufacturing systems in batch productions has been much developed aimed at realizing flexible small volume batch productions. The control structures of the manufacturing systems developed, such as FMS (Flexible Manufacturing System) and FMC (Flexible Manufacturing Cell), are generally hierarchical. The hierarchical control structure is suitable for economical and efficient batch productions in steady state, but not adaptable to very small batch productions with dynamic changes in the volumes and the varieties of the products.

Computer systems and manufacturing cell controllers have recently made much progress, and individual computers and controllers are now able to share the decision making capabilities in the manufacturing systems. The network architectures are widely utilized for the information exchange in the design and the manufacturing.

New distributed architectures of manufacturing systems are therefore proposed to realize more flexible control structures of the manufacturing systems, which are adaptable to the dynamic changes in the volume and the variety of the products and also the unforeseen disruptions, such as malfunction of manufacturing equipment and interruption by high priority jobs. They are so called as ADMS (Autonomous Distributed Manufacturing Systems) [1], BMS (Biological Manufacturing Systems) [2], and HMS (Holonic Manufacturing Systems) [3] [4] [5] [6].

Distributed scheduling methods were proposed and applied to the real-time production scheduling problems of the HMS, in the previous research [4]. The proposed method was adaptable to the dynamic changes and the unforeseen disruptions, and it was suitable for the improvement of the objective functions of the whole HMS such as total make span. However, there were still remaining scheduling problems from the viewpoint for the improvement of the objective functions of the individual components of the HMS.

Therefore, a real-time scheduling method based on the utility values have been proposed and applied to the HMS, in order to improve the objective function values of the individual components of the HMS [5]. The holons in the HMS are divided into three classes based on their roles in the manufacturing processes and the scheduling processes.

(a) Resource holons: They transform the job holons in the manufacturing process. In the scheduling process, they evaluate the utility values for the candidate job holons which are processed by the resource holons in the next time period.
(b) Job holons: They are transformed by the resource holons from the blank materials to the final products in the manufacturing process. In the scheduling process, they evaluate the utility values for the candidate resource holons which carry out the machining operations in the next time period.
(c) Coordination holon: It selects a most suitable combination of the resource holons and the job holons for the machining operations in the next time period, based on the utility values sent from the resource holons and the job holons.

Multi-agent reinforcement learning is newly proposed and implemented to the job holons and resource holons, in order to improve their coordination processes. A reinforcement learning method was proposed and applied to centralized scheduling problems for semiconductor manufacturing processes [7]. In the present research, a reinforcement learning is applied to the agent-based distributed scheduling processes for the manufacturing processes of machine products.

2 Real-Time Scheduling Method for HMS [5]

2.1 Information for Real-Time Scheduling

It is assumed here that the individual job holons have the following technological information.

M_{ik}	: k-th machining operation of the job holon i. ($i = 1,..,\alpha$), ($k = 1,..,\beta$).
AC_{ik}	: Required machining accuracy of machining operation M_{ik}. It is assumed that the machining accuracy is represented by the levels of accuracy indicated by 1, 2, and 3, which mean rough, medium high, and high accuracy, individually.

R_{ikm}	: m-th candidate of resource holon, which can carry out the machining operation M_{ik}. ($m = 1,...,\gamma$).
W_i	: Waiting time until the job holon i becomes idle if it is under machining status.

The individual resource holons have the following technological information.

T_{ikm}	: Machining time in the case where the resource holon R_{ikm} carries out the machining operation M_{ik}.
MAC_{ikm}	: Machining accuracy in the case where the resource holon R_{ikm} carries out the machining operation M_{ik}. MAC_{ikm} is also represented by the levels of 1, 2 and 3.
MCO_{ikm}	: Machining cost in the case where the resource holon R_{ikm} carries out the machining operation M_{ik}.
W_{ikm}	: Waiting time until resource holon R_{ikm} becomes idle if it is under machining status.

2.2 Real-Time Scheduling Process Based on Utility Values

A real-time scheduling process based on the utility values have been proposed, in the previous research [5], to select a suitable combination of the job holons and the resource holons which carries out the machining operation in the next time period.

At the time t, all the 'idling' holons have to select their machining schedules in the next time period. The following procedure is proposed for the individual holons to select their machining schedules.

(1) Retrieval of status data
The individual 'idling' holons firstly get the status data from the other holons which are 'operating' or 'idling'. The 'idling' holons can start the machining operation in the next time period.

(2) Selection of candidate holons
The individual 'idling' holons select all the candidate holons for the machining operations in the next time period. For instances, the job holon i selects the resource holons which can carry out the next machining operation M_{ik}. On the other hand, the resource holon j select all the candidate job holons which can be machined by the resource holon j.

(3) Determination of utility values
The individual 'idling' holons determine the utility values for the individual candidates selected in the second step. For instances, the job holon determines the utility values, based on its own decision criteria for all the candidate resource holons which can carry out the next machining operation. The utility values are given as follows.

$JUV_i(j)$ $(0 \leq JUV_i(j) \leq 1)$: Utility value of the candidate resource j for the job holon i.
$RUV_j(i)$ $(0 \leq RUV_j(i) \leq 1)$: Utility value of the candidate job i for the resource holon j.

(4) Coordination
All the 'idling' holons send the selected candidates and the utility values of the candidates to the coordination holon. The coordination holon determine a suitable combination of the job holons and the resource holons which carry out the machining operations in the next time period, based on the utility values. The decision criteria of the coordination holon is to maximize the total sum of the utility values of all the holons.

Table 1. Objective functions of holons

Objective functions	Objective function values
Efficiency of resource holon	Σ Machining time / Total time
Machining accuracy of resource holon	Σ(Machining accuracy of resources – Required machining accuracy of jobs)
Flow-time of job holon	Σ(Machining time + Waiting time)
Machining cost of job holon	Σ(Machining cost of resources)

2.3 Evaluation of Utility Values

The utility values are evaluated based on the decision criteria of the individual holons, and various decision criteria are considered for the holons. Therefore, it is assumed that the individual holons have one of the objective functions shown in Table 1 for evaluating the utility values.

The following procedures are provided for the resource holons to evaluate the utility values. Let us consider a resource holon j at a time t. It is assumed that $TT_{j\cdot t}$, $ME_{j\cdot t}$, and $MA_{j\cdot t}$ show the total time after the resource holon j starts its operations, the efficiency, and the evaluated value of machining accuracy of the resource holon j, respectively. If the resource holon j selects a candidate job holon i for carrying out the machining operation M_{ik}, the efficiency and the evaluated value of the machining accuracy are estimated by the following equations.

$$ME_{j\cdot t+1}(i) = (ME_{j\cdot t} \cdot TT_{j\cdot t} + T_{ikj}) / (TT_{j\cdot t} + T_{ikj} + W_i) . \quad (1)$$

$$MA_{j\cdot t+1}(i) = MA_{j\cdot t} + (MAC_{ikj} - AC_{ik}) . \quad (2)$$

where, the resource holon j can carry out the machining operation M_{ik} of job holon i ($j = R_{ikm}$).

As regards the job holons, the following equations are applied to evaluate the flow-time and the machining costs, for the case where a job holon i selects a candidate resource holon j ($= R_{ikm}$) for carrying out the machining operation M_{ik}. It is assumed that $JT_{i\cdot t}$ and $JC_{i\cdot t}$ give the total time after the job holon i is inputted to the HMS and the machining cost, respectively.

$$JT_{i\cdot t+1}(j) = JT_{i\cdot t} + T_{ikj} + W_{ikj} . \quad (3)$$

$$JC_{i\cdot t+1}(j) = JC_{i\cdot t} + MCO_{ikj} . \quad (4)$$

The objective functions mentioned above have different units. Some of them shall be maximized and others shall be minimized. Therefore, the utility values are normalized from 0 to 1.

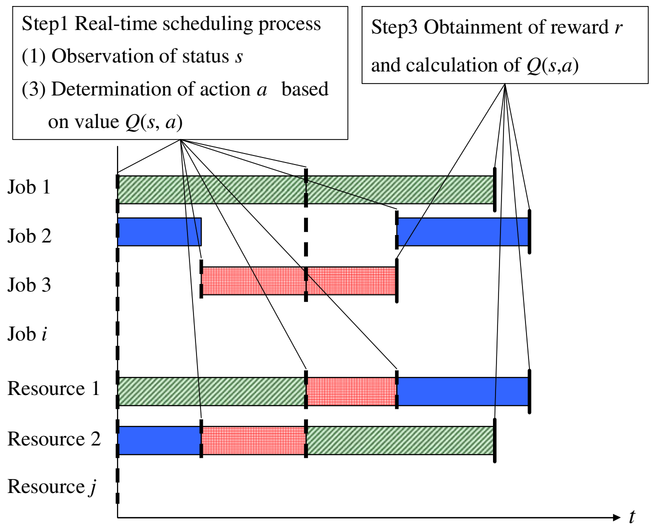

Step1 Real-time scheduling process
(1) Observation of status s
(3) Determination of action a based on value $Q(s, a)$
Step3 Obtainment of reward r and calculation of $Q(s,a)$
Job 1
Job 2
Job 3
Job i
Resource 1
Resource 2
Resource j
t

Fig. 1. Application of multi-agent reinforcement learning

3 Application of Multi-agent Reinforcement Learning

A multi-agent reinforcement learning is newly proposed and implemented to the job holons and resource holons, in the present research, in order to improve their coordination processes. In the reinforcement learning method [8], an agent must be able to sense the status of the environment to some extent and must be able to take actions that affect the status. The agent also must have a goal or goals relating to the status of the environment.

Figure 1 summarizes the multi-agent reinforcement learning procedure proposed here. The individual job holons and resource holons carry out the following four steps to obtain their suitable decision criteria for evaluation of the utility values by applying the multi-agent reinforcement learning.

Step1. The individual job holos and resource holons carry out the real-time scheduling process described in section 2.2, when their previous machining operations are finished. The real-time scheduling process (1) and (3) are modified as following for implementation of the multi-agent reinforcement learning.

(1) Retrieval of status data

The individual 'idling' holons get the status data from the other holons which are 'operating' or 'idling', and observe the status s of the manufacturing systems.

(3) Determination of utility values

The individual job holons and resource holons execute the action a based on the value $Q(s, a)$, to evaluate the utility values for all the candidate machining operations in the next time period.

Where, s and a represent the status and the actions in the reinforcement learning method, respectively.

Step2. The real-time scheduling process are repeated until all the machining operations of the job holons are finished by the resource holons in the HMS.

Step3. The individual job holons and resource holons obtain the reward r based on their own objective function values, and calculate the value $Q(s, a)$.

Step4. Step1 to Step3 are repeated for the new job holons to be manufactured in the manufacturing systems, in order to converge the value $Q(s, a)$ of the individual job holons and resource holons.

In these steps, the status s, the action a and the reward r are given as follows.

(1) Status s

The status s observed by the job holons and the resource holons is represented by the following equation, in the present research.

$$s = (s_1, s_2, s_3, s_4)\ . \tag{5}$$

where, s_p (p =1, 2, 3, 4) are the number of 'idling' holons, that have the objective functions of efficiency, machining accuracy, flow-time, and machining cost, respectively. This means that the learning process of the individual holons are carried out based on the numbers and the types of the 'idling' holons.

(2) Action a

The individual job holons and resource holons select the parameter n (= 1/5, 1/3, 1, 3, or 5) in the following equation to evaluate the utility values.

$$UV' = (UV)^n\ . \tag{6}$$

where, UV is the utility value calculated by the individual job holons and resource holons described in section 2.3. UV' is the modified utility value by applying the action a based on the status s.

ε-greedy method [8] is applied for the individual job holons and resource holons to determine the action a.

(3) Reward r

The individual job holons and resource holons obtain the reward r based on their own objection values. Three different methods are considered to calculate the reward r.

Type 1. Reward calculated by the objective function values of individual holons

The individual job holons and resource holons obtain the reward r_h given by following equations, based on their own objective functions.

(a) For the case that the objective function is efficiency

$$r_h = (a_h - b_h) / b_h\ . \tag{7}$$

(b) For the case that the objective function is either machining accuracy, flow time or machining cost

$$r_h = (b_h - a_h) / b_h\ . \tag{8}$$

where, a_h and b_h are the objective function values obtained by applying the proposed method with the reinforcement learning, and ones obtained without the reinforcement learning.

Type 2. Reward calculated by the objective function values of holons which have same objective function

The individual job holons and resource holons obtain the reward r_p given by following equations.

$$r_p = \sum_{h=1}^{\tau} r_h / \tau . \tag{9}$$

where, p and τ are the types of objective functions and the total number of holons with p-th type of objective functions, respectively. r_h is calculated by Eq. (7) and (8) based on the types of objective functions.

Type 3. Reward calculated by the objective function values of all holons
The individual job holons and resource holons obtain the reward r_q given by following equations.

$$r_q = (1/4) \sum_{p=1}^{4} r_p . \tag{10}$$

where, r_p is calculated by Eq. (9).

The value $Q(s, a)$ is determined by applying the monte carlo method [8]. The individual job holons and resource holons save the n rules (s_t, a_t) $(t = 0, 1, \ldots, n\text{-}1)$ between the time when they obtain the reward r and the time when they obtain the new reward r. The rule (s, a) means the set of status s and action a. The value $Q(s, a)$ is calculated by the following equations.

$$SumReward(s_t, a_t) \leftarrow SumReward(s_t, a_t) + r . \tag{11}$$

$$Q(s, a) \leftarrow SumReward(s, a) / RewardCount . \tag{12}$$

where, $SumReward(s, a)$ is the cumulative rewards in the case where the action a is applied in the status s. $RewardCount$ is the total number in the case where the rule (s, a) get the reward r.

4 Case Study

Some case studies have been carried out to verify the effectiveness of the proposed methods. The HMS model considered in the case studies has 10 resource holons. The individual resource holons have the different objective functions and the different machining capacities, such as the machining time T_{ikm}, the machining accuracy MAC_{ikm}, and the machining cost MCO_{ikm}.

As regards the job holons, 3 cases are considered in the case study, which have 16 job holons, 20 job holons and 30 job holons. The individual job holons have the different objective functions and the machining sequences. It is assumed that the same job holons are inputted to the HMS after the resource holons finish all the manufacturing processes. 12 cases are considered, in the case study, by changing the machining capacities of the resource holons.

ε is set to 0.2 for the ε-greedy method.

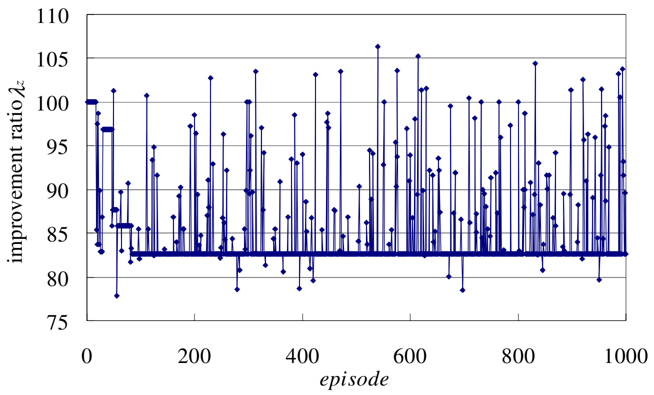


Fig. 2. Improvement ratio

Figure 2 shows the best result for the case where the reward is calculated by using Type3 described in section 3. In the figure, the horizontal and vertical axes show the *episode* and the improvement ratio λ_z, respectively. The *episode* means here the number of repetitions of all the manufacturing processes of the inputted jobs. The improvement ratio λ_z means the ratio between the objective function values of all the holons obtained by the proposed method and the ones by the conventional method in the case z. λ_z is calculated by following equation.

$$\lambda_z = \sum_{h=1}^{\nu} \mu_h / \nu \,. \tag{13}$$

where, μ_h and ν are the improvement ratio of the objective function values of the holon h and the total number of holons, respectively. The μ_h is calculated by the following equation based on the type of the objective functions.

(a) For the case that the objective function is efficiency

$$\mu_h = b_h / a_h \,. \tag{14}$$

(b) For the case that the objective function is either machining accuracy, flow time or machining cost

$$\mu_h = a_h / b_h \,. \tag{15}$$

where, a_h and b_h are the objective function values of the individual holons h obtained by the proposed method and the previous conventional method. As shown in the figure, the improvement ratio λ_z is converged until the episode reaches to 100.

Figure 3 shows the average improvement ratio $\lambda average_z$ of the best case and the worst case. Following equation gives the $\lambda average_z$ which means the average of improvement ratio λ_z until the *episode* reaches to ω in the case z.

$$\lambda average_z = \sum_{episode=1}^{\omega} \lambda_{episode} / \omega \,. \tag{16}$$

where, $\lambda_{episode}$ is the improvement ratio λ_z at the *episode*.

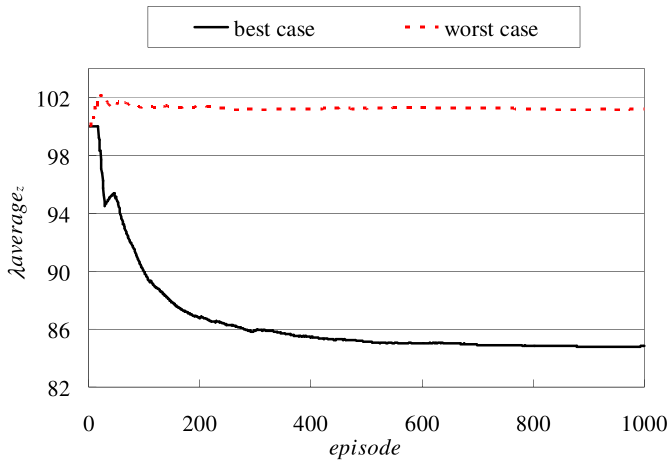


Fig. 3. Average improvement ratio

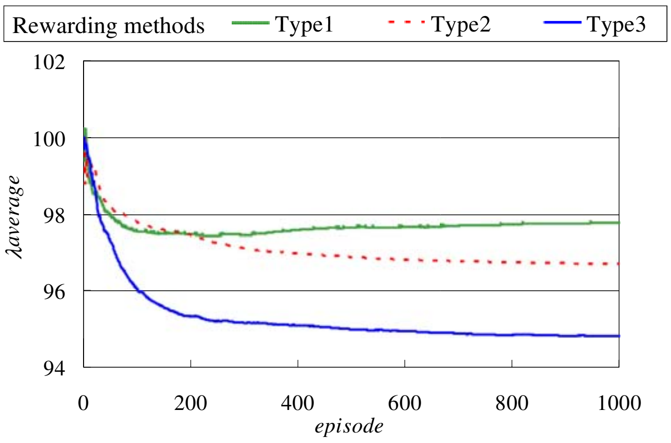


Fig. 4. Comparison of rewarding methods

Figure 4 shows the comparison of the cases using Type1, Type2 and Type3 rewarding methods described in section 3, from the view point of *λaverge*. *λaverge* means the average of $\lambda average_z$ in the all 12 cases. *λaverge* is calculated by the following equation.

$$\lambda average = \sum_{z=1}^{12} \lambda average_z / 12. \tag{17}$$

As shown in Fig. 4, all cases are effective to improve the objective function values in comparison with previous method without reinforcement learning. It means that the individual job holons and resource holons obtain the suitable decision criteria for evaluation of utility values. However, as shown in the figure, the value $Q(s, a)$ does not converge in the case using Type1 where the reward is calculated by the objective function values of individual holons. The individual job holons and resource holons most improve their objective function values in the case using Type3 where the reward is calculated by the objective function values of all holons.

5 Conclusions

New systematic methods are proposed here to improve the coordination process among the job holons and the resource holons based on the multi-agent reinforcement learning. The following remarks are concluded.

(1) The real-time scheduling process are modified for implementation of multi-agent reinforcement learning in order to obtain the suitable decision criteria for evaluation of utility values.
(2) The status, the action and the reward are defined for the individual job holons and the resource holons to evaluate the suitable utility values based on the status of the HMS.
(3) Some case studies of the real-time scheduling have been carried out to verify the effectiveness of the proposed methods in comparison with the previous method. It was shown, through case studies, that the proposed methods are effective to improve the objective function values of the individual holons. The objective function values of individual holons are improved most effectively in the case where the reward is calculated by the objective function values of all holons.

References

1. Moriwaki, T., Sugimura, N.: Object-oriented modeling of autonomous distributed manufacturing system and its application to real-time scheduling. In: Proc. of the ICOOMS 1992, pp. 207–212 (1992)
2. Ueda, K.: An approach to bionic manufacturing systems based on DNA-type information. In: Proc. of the ICOOMS 1992, pp. 303–308 (1992)
3. Hendrik, B., Jo, W., Paul, V., Luc, B., Patrick, P.: Reference architecture for holonic manufacturing systems: PROSA. Computers in Industry 37, 255–274 (1998)
4. Sugimura, N., Tanimizu, Y., Iwamura, K.: A Study on real-time scheduling for holonic manufacturing system. CIRP journal of manufacturing systems 33(5), 467–475 (2004)
5. Iwamura, K., Okubo, N., Tanimizu, Y., Sugimura, N.: Real-time scheduling for holonic manufacturing systems based on estimation of future status. International journal of production research 44(18-19), 3657–3675 (2006)
6. Iwamura, K., Nakano, A., Tanimizu, Y., Sugimura, N.: A study on real-time scheduling for holonic manufacturing systems – simulation for estimation of future status by individual holons. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 205–214. Springer, Heidelberg (2007)
7. Fujii, N., Takasu, R., Kobayashi, M., Ueda, K.: Reinforcement learning based product dispatching scheduling in a semiconductor manufacturing system. In: Proc. of the 38th CIRP International seminar on manufacturing systems, CD-ROM (2005)
8. Sutton, R., Barto, A.: Reinforcement learning: An introduction. MIT Press, Cambridge (1998)

An Open-Control Concept for a Holonic Multiagent System

Emmanuel Adam, Thierry Berger, Yves Sallez, and Damien Trentesaux

Univ Lille Nord de France, F-59000 Lille, France
UVHC, LAMIH, F-59313 Valenciennes, France
CNRS, UMR 8530, F-59313 Valenciennes, France
{emmanuel.adam,thierry.berger,yves.sallez, damien.trentesaux}@univ-valenciennes.fr
http://www.univ-valenciennes.fr/LAMIH/

Abstract. MAS are particularly adapted to deal with distributed and dynamic environment. The management of business workflow, or data flow, flexible manufacturing systems is typically a good application field for them. This kind of application requires centralization of the data control and flexibility to face with changes on the network. In the context of FMS, where products and resources entities can be seen as active, this paper presents the open-control concept and gives an example of its instantiation with holonic scheme. The open-control concept proposed in this paper exhibits the classic explicit control, as well as an innovative type of control called implicit control that allows system entities to be influenced via an Optimization Mechanism (OM). We illustrate our proposition by an implementation on a flexible assembly cell in our university.

Keywords: open-control, flexible manufacturing systems, holonic manufacturing system.

1 Introduction

To be competitive, manufacturing should adapt to changing conditions imposed by the market. The greater variety of products, the possible large fluctuations in demand, the shorter lifecycle of products expressed by a higher dynamics of new products, and the increased customer expectations in terms of quality and delivery time are challenges that manufacturing companies have to deal with to remain competitive. Besides these market-based challenges, manufacturing firms also need constantly to be flexible and adapt to newly developed processes and technologies and to rapidly changing environmental protection regulations.

In recent decades, scientific developments in the field of production have defined new architectures including the heterarchical/non-hierarchical architectures that play a prominent role in FMS.

Holonic Systems are typically one of these new architectures, and some works have shown its efficiency.

A holonic organisation has specific working mechanisms, described initially in [1]. One of the particularity of holonic concept is the notion of control, which is both distributed, and partly centralized or managed at a higher level. This typical control is particularly interesting in the management of systems evolving in dynamic and demanding environment.

This paper presents our motivation regarding the concept of open-control, then briefly the key concept that we use to build our Holonic multiagent systems (section 3), and next our point of view of the open-control concept and the reasons for developing it (sections 4). An instantiation of this concept using the holonic manufacturing approach is then presented (section 5). This concept is an extension of a previous work in the domain of heterarchical control and includes the concept of implicit control in addition to the traditional explicit control.

2 Motivations

In this paper, the term "control" includes what is generally accepted as the whole loop, from sensors to actuators. As a result, a closed loop can then be identified as something that exists between a system that controls and a system that is controlled [2].

Traditional approach is mainly associated to the initial CIM (Computer Integrated Manufacturing) concept and usually leads to centralized or hierarchical control structures. Due to the complexity of manufacturing problems, the usual practice has been to split the overall problem into hierarchically-dependent functions that operate within decreasing time-ranges, such as planning, scheduling and/or monitoring. This traditional approach is known to provide near optimal solutions, but only when hard assumptions are met, for example, no external (e.g., urgent orders) or internal (e.g., machine breakdowns) perturbations, well-known demands, and/or supplier reliability. Since reality is rarely so deterministic, this approach rapidly becomes inefficient when the system must deal with stochastic behavior.

The above observations have led researchers to define a second approach to designing control architectures. These control architectures, also called emergent or self-organized, can be categorized in four types [3]: bionic et bio-inspired, as proposed by Okino [4] and Dorigo et Stutzle [5]; multi-agent, as proposed by Maione et Naso [6]; holonic, as proposed by Van Brussel et al. [7]; and heterarchical, as proposed by Trentesaux et al. [8]. The expected advantages of such architectures are related to agility: in the short term, such architectures are reactive and in the long term, they are able to adapt to their environment. However, these last control architectures suffer from the lack of long-term optimality, even when the environment remains deterministic, which can be called "myopic" behavior. This is the main reason why such control architectures are not really used by industrialists at the moment.

The aim of this paper is to propose a global control concept, called "open-control", in which traditional control is augmented by a new kind of control:

"implicit control". In this concept, entities can be strictly controlled hierarchically and, at the same time, they can be influenced heterarchically by their environment and/or by other entities. This concept would make it possible to design control systems that are both agile and globally optimized, thus reducing the myopic behavior of self-organized architectures and increasing the agility of traditional architectures. Combining the two types of control in the same architecture creates new challenges since the two types of control must now be managed and integrated within the larger control concept.

This paper describe our "open-control" concept that is particularly suited to HoloMAS that are defined in the following section.

3 Proposition of a HoloMAS Using Roles

If holonic model is not suitable for all the organisations, it is particularly adapted to an organisation having a flexible and hierarchic architecture (whose parts have a relative autonomy), where notion of roles, responsibilities and cooperation are important.

We have analysed holonic rules proposed in [1] and we proposed a first set of formal definitions of a holonic multi-agent system, [9] having a pyramidal architecture and whose agents have to reach global objective of the system while having a personal goal to reach. We present in this article a subset, relative to the notion of control, of these definitions.

World definition: We define world as being constituted of the MAS and the environment:
$world = (environment, mas)$

Environment definition: This environment is constituted of objects:
$environment : E = \{object_0, object_1, ..., object_n\}$
An object has a name, some states, properties and relations with other objects; when a property is activated, it can modify the states or activate other properties of this object or of its relation. An object has no goal.

MAS and agent definition: MAS is simply defined as a set of agents. Each agent is composed of: states; knowledge (social (name of the acquaintances), environmental and personal (which contains the agent goal and agent properties dependent on the application)); messages; a perception function; behaviour rules (functions modifying agent states according to current states, knowledge and received messages in order to reach the collective goal while moving toward its personal goals); a dynamic list of roles that define some behaviors relatively to the application. Indeed, an agent can: receive a role; leave a role; delegate a role or goal to its acquaintances (our principle of delegation is inspired from the works of [10]).

Roles and rules definition: Notion of roles is now classically admitted for an agent (see for example [11], [12]). In our case, we consider a role as: a set of knowledge (environmental and social); and a set of rules that allows it to to reach its goals. So, the definition of a role R is, for us, relatively similar to agent definition.

A role contains classicaly some prerequirements (in its personal knowledge, to allow to define dependencies between roles[1]) that must have agents that will play it, and some consequences for the agents that embody it.

Role contains also its workload for the agent that will take it, and agents have maximum workload thresholds (mwt). The (mwt) of an agent decreases with the depth and can not go under the limit 1 (this limit can be higher according to the study case); holon having a minimal mwt is said a leaf holon or an atomic holon.

Prerequirements and workloads of roles affect thus theirs distribution on a HoloMAS.

The agent behaviour consists in acting as long as the collective goal and the individual goal of an agent and of its roles are not reached; namely, as long as the states of the agent and of its roles does not correspond to the constraints in the Individual Goal and the Collective Goal. For that, the agent chooses actions in the rules of its roles (cf alg. 1), and in its own rules, with respect to the collective goal. There is an implicit societal control here in the sense that a rule is chosen only if it is *compatible* to the collective goal; and an implicit environmental control when the agent chooses and schedules its rules according to its perception.

A rule is chosen relatively to the intended goal and according to the priority (utility level) of the rule. These priority is either defined a priori, or computed from heuristics, or infered from the last use of the rule (the rule priority increases, according to its capacity to go near the agent's goal, each time the agent chooses it to reach its goal).

Algorithm 1. Rule choosing of an agent a

```
function CHOOSEROLERULE($Knowledge$, $EnvironmentPerception$, $messages_a$)
    for all $role \in agent.roles$ do
        if ACTIVABLE($role$, $EnvironmentPerception$) then
            SORTBYPRIORITY($Role.rules$, $EnvironmentPerception$)
            for all $rule \in Role.rules$ do
                if COMPATIBLE($rule$, $CollectiveGoal$) then return rule
                end if
            end for
        end if
    end for
end function
```

[1] This dependency relation is reflexive and transitive. It is the first step of our representation of the dependency; we plan to use a more complete role dependency definition like the one proposed in [13] that describe dependency between roles relatively to an objective.

If the agent are able to control their activities to avoid violation of the collective goal, the notion of open-control is not really considered; so we propose a concept to deal with this particular control.

4 The Open-Control Concept

4.1 General Framework

We propose a control framework in which an entity can not only achieve its goal in terms of the system objectives but also in terms of its own objectives. An entity can be a resource or an active product. An active product is an entity that able to inform, communicate, decide and act in order to reach its goals in solving resource allocation and routing problems (for more details on the typology and advantages of active products, see [14]).

This framework has various levels: level i serves the level $i + 1$ above it and is served by level $i - 1$ below it. Long term problems are solved at the top and real-time problems are solved at the bottom. In this framework, autonomous entities can exist in both the physical and informational worlds.

In typical Flexible Manufacturing Systems (FMS), the physical world is made up of the physical parts of the entities, which range from passive products (e.g., a pure raw product) to a resource's raw materials (e.g., the mechanical structure of an assembly robot) and, by extension, to the Human Operator's body, which is the special case of a non-artificial entity. The informational world is usually composed of 3 control levels (1 to 3): *strategic problem* solving at the top (level 3), *operational problem* solving at the bottom (level 1), and *tactical problem* solving in the middle (level 2). For example, in classical FMS, ERP (Enterprise Resource Planning) would be on level 3, MES (Manufacturing Execution System) would be on level 2, and Automation et PLC (Programmable Logic Controller) would be on level 1.

In our framework, each autonomous entities of a system is immersed in an informational level orchestrated by an Optimization Mechanism (OM), and each entity is always trying to achieve its own objective through decision-making that is influenced by either a societal or an environmental OM.

4.2 Implicit and Explicit Control

Before describing in detail the explicit and implicit control used in our concept, we must first introduce a model of an advanced FMS control based on the framework shown in Figure 1.

Figure 1 represents an advanced FMS in which entities (e.g., resources, active products) have decisional capacities (represented by D in a circle) and thus play an active role in achieving their goals.

In Figure 1, the two main levels of our framework are represented: the controlled level i, in which the different entities evolve, and the controller level $i+1$, which controls level i. The physical world is also represented through the physical base of the different entities.

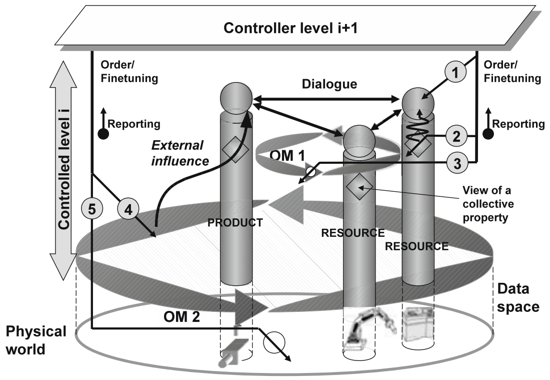


Fig. 1. Description of the Advanced FMS control model

As mentioned above, each entity must meet its own goal but in terms of collective global performance criteria. At their individual levels, the entities all have a self-made view (personal knowledge) of the collective performance criteria (represented in Figure 1 by a small diamond). This partial view is achieved by dialogue among entities. These exchanges will support a mechanism for optimizing collective performance. We define three types of control:

Explicit control. Explicit control involves a classic top-down, master-slave approach to control: a controller sends orders to a specific clearly identified subordinate (arrow n^o1 in Figure 1). This subordinate sends information in return.

Implicit control via a societal OM. This kind of implicit control can be performed in two ways. The first involves finetuning the partial view of a collective property inside an entity (arrow n^o2 in Figure 1). This corresponds to the method *Compatible* in Algorithm 1. This modification can be seen as an internal influence that modifies the entity's behavior. This behavioral modification then influences the other entities via the societal optimization mechanism, which is supported by dialogue. The second way involves changing the dynamics of the dialogue in the societal optimization mechanism (arrow n^o3 in Figure 1) by modifying the dialogue parameters inside the entity. This second way has a direct impact on the overall collective performance.

Implicit control via an environmental OM. This kind of implicit control is performed via the informational environment in two ways: the first (arrow n^o4 in Figure 1) involves acting on the data directly (e.g., creating, updating, erasing), while the second (arrow n^o5 in Figure 1) involves finetuning the parameters used by the environmental optimization mechanism. Both actions generate an external influence that can affect all that entities able to access the informational environment. For this type of implicit control, no communication between the entities is required. The methods *Activable* and *sortByPriority* in Algorithm 1 are linked to this control.

The main difference between implicit and explicit control is that, in implicit control, the final entity is not directly targeted. Implicit control uses a dedicated intermediate, which does not directly target the influenced entities. This type of control is very interesting for applications which involve multiple entities that can not be controlled directly.

5 Application to Usual FMS Control Architectures

In this section, we describe our experimentation with an active product and report our results. This active product (AP) is able to communicate, adapt itself to perturbations, and make decisions in order to reach an assigned goal. The AP operates in a heterarchical architecture without any central control entity. In our experiments, we focused on the dynamic FMS routing of two kinds of products — passive and active — from a source node to a destination node.

5.1 Experimental Context

The experimentation was performed in a flexible assembly cell at the Valenciennes AIP-Primeca pole. This cell is composed of seven W_i workstations placed around a flexible conveyor system based on Montrac technology [15]. This system is a monorail transport system using self-propelled shuttles to transport materials on tracks. Figure 2 focuses on the right part of the cell (labeled target area), which has eleven nodes:

– $N1$, $N4$, $N6$, $N8$ and $N11$ (in white) are nodes/stations where services can be obtained; they constitute the possible destination nodes.
– $N2$, $N3$, $N5$, $N7$, $N9$ and $N10$ (in gray) are divergent transfer gates, used to direct shuttles on the chosen tracks. The routing decision must be made based on the information available at these locations.

The other transfer gates are not considered. They are only used to join convergent tracks, and thus no decision-making is required.

In our implementation, the passive product was transformed into an active product AP by adding an embedded device to this product. This product is based on a personal digital assistant (a PDA with an Intel processor, IrDA communications and Windows Mobile operating system).

A routing decision must be made at each node, which is composed of a node controller and two data communication systems. The node controller 750-841 [16] oversees the transfer gate, helping to avoid collisions.

If necessary, this node controller also updates the AP's routing graph — a topological graph of the cell valued with the travel times on each arc.

The data communication systems consist of one Ethernet system for node-to-node interaction and one IrDA system for node-to-AP communication esb-101

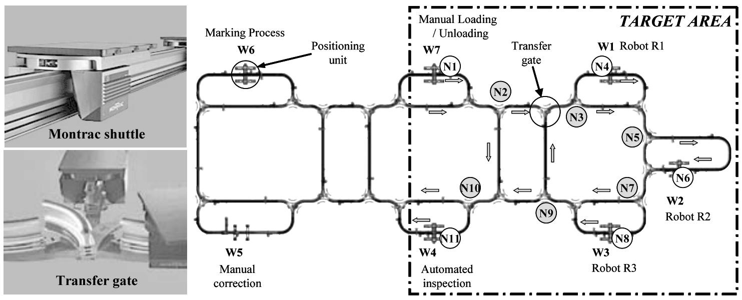


Fig. 2. The flexible assembly cell

[17]. We chose to use infra-red technology for its low energy consumption, its high transfer rates (more than 100 Kb/sec) and its short range, which enables geographical node localization.

5.2 AP Routing Algorithm

In our experiments, $N1$ and $N11$, respectively, were the source and destination nodes for the AP routing. In each Active Product, a role use a Dijkstra's algorithm [18] to elaborate the best path (i.e., the path with the shortest travel time) using the arc valuations from the routing graph step by step on each node. For passive products, only a static routing graph could be used, and such products were routed according to static rules. Dijkstra's algorithm was activated once at the beginning of the experiment, and the results were used, whatever they led to. Figure 3 illustrates the different cases studied, and shows different routing scenarios.

Scenario *S1* was the reference case. A static routing graph was used, and no perturbation was introduced. Scenario *S2* maintained the static routing graph but introduced a perturbation (e.g., bottleneck, slow down, break down) that affected the fluidity of the path $N2 - N10$ at time τ. Scenario *S3* involved a dynamic routing graph, but maintained the same perturbation as in scenario *S2*. For the reference case with static data, the best path obtained by Dijkstra's algorithm was $[N1 - N2 - N10 - N11]$. These passive products travelled this path

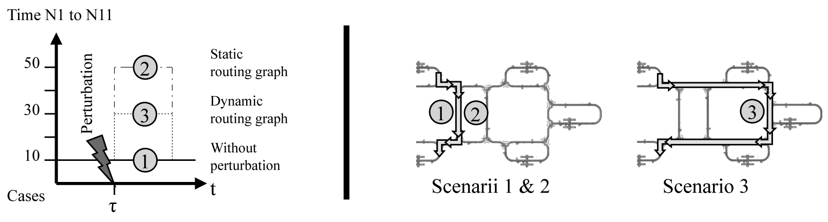


Fig. 3. Test cases and scenarii

in 10 time units. We then analyzed the results of scenario *S2* and *S3*, illustrating the behavior of a passive product and an AP faced with a perturbation. In scenario *S2*, the routing graph is static and thus does not include the new $N2 - N10$ arc value. Consequently, passive products systemically took the previously calculated path, altering the efficiency of the routing process and increasing the travel time between $N1$ and $N11$ from 10 to 50 time units. In scenario *S3*, the routing graph of each node controller was updated with the measured time between $N2$ and $N10$. Exploiting this new graph valuation, the AP elaborated an new updated shortest path $[N1 - N2 - N3 - N5 - N7 - N9 - N10 - N11]$ with the Dijskstra algorithm. As a result, after a perturbation, only the first affected AP was delayed; the other APs knew that there was a perturbation on arc $N2 - N10$, and they chose the alternative path to reach their destination. The APs took the path $[N1 - N2 - N10 - N11]$ again when an external event informed the nodes controllers that the perturbation had been resolved.

5.3 Conclusions Drawn from the Experiment

In terms of control, the proposed system has the following advantages:

- the APs are able to find an efficient alternative path more easily with dynamic routing approach, making the system more reactive;
- local decision-making without any central point of control makes the system more robust against perturbations;
- adding a new node means adding a controller, an IrDA beam and connecting them to the network, thus improving the system's scalability; and
- since each AP can be precisely localized with the IrDA beams, there is no need for a global AP localization system.

We did, however, note one drawback that is linked to using the PDA: the IrDA connection is slow, with a total connection and disconnection time of approximatively 5 seconds.

6 Conclusions

In this paper, we have presented our approach to define holonic multiagent systems, and an open-control concept. In addition, the designed holonic architecture has also been validated on a real production cell to verify its consistency. Of course, a complete validation procedure must be realized (comparison with other control architectures on a benchmark?), but this is in fact beyond the scope of this paper. Short term perspective concerns then the validation of the open control concept on other classes of control architectures (CIM, MAS, bio-inspired?). Concerning middle term perspectives, we are now working on a design method that will help designers to specify their open-control architecture models exhaustively, thus informing them in advance all the properties inherited from his design choices and the set of commonly encountered issues to be solved; and we are going to work on more precise but general specifications of the implicit controls.

Acknowledgments. The present research work has been supported by International Campus on Safety and Intermodality in Transportation the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the Ministry of Higher Education and Research, the ANR (National Agency for Research) TTT and Robot, and the National Center for Scientific Research. The authorsgratefully acknowledge the support of these institutions.

References

1. Koestler, A.: The Ghost in the Machine. Arkana Books (1969)
2. Wiener, N.: Cybernetics: Or Control and Communication in Animal and the Machine. MIT Press, Cambridge (2000)
3. Bousbia, S., Trentesaux, D.: Self-organization in distributed manufacturing control: state-of-the-art and future trends. In: 2002 IEEE International Conference on Systems, Man and Cybernetics, vol. 5, p. 6 (2002)
4. Okino, N.: Bionic manufacturing system. In: Peklenik, J. (ed.) CIRP, Flexible Manufacturing Systems: Past-Present-Future, pp. 73–95 (1993)
5. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
6. Maione, G., Naso, D.: A soft computing approach for task contracting in multi-agent manufacturing control. Computers in Industry 52(3), 199–219 (2003); Soft Computing in Industrial Applications
7. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems: Prosa. Comput. Ind. 37(3), 255–274 (1998)
8. Trentesaux, D., Dindeleux, R., Tahon, C.: A multicriteria decision support system for dynamic task allocation in a distributed production activity control structure. International Journal of Computer Integrated Manufacturing 11(1), 3–15 (1998)
9. Adam, E., Grislin-Le Strugeon, E., Mandiau, R.: Flexible hierarchical organisation of role based agents. In: 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 186–191. IEEE, Los Alamitos (2008)
10. Saidani, O., Nurcan, S.: A role based approach for modeling flexible business processes. In: The 7th Workshop on Business Process Modelling, Development, and Support (BPMDS 2006), pp. 111–120. Springer, Heidelberg (2006)
11. Wooldridge, M., Jennings, N., Kinny, D.: The gaia methodology for agent oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems (2000)
12. Mathieu, P., Routier, J.C., Secq, Y.: RIO: Roles, interactions and organizations. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS, vol. 2691, p. 147. Springer, Heidelberg (2003)
13. Dignum, V., Dignum, F.: Coordinating tasks in agent organizations. In: Dunin-Keplicz, B., Omicini, A., Padget, J.A. (eds.) EUMAS. CEUR Workshop Proceedings, vol. 223, CEUR-WS.org (2006)
14. Zbib, N., Raileanu, S., Sallez, Y., Berger, T., Trentesaux, D.: From passive products to intelligent products: the augmentation module concept. In: 5th International Conference on Digital Enterprise Technology, Nantes, France (October 2008)
15. Montech-Conveyor-systems (2007), http://www.montech.com
16. Wago-system, i.c. (2007), http://www.wago.com
17. Clarinet System, N.C. f. (2007), http://www.clarinetsys.com
18. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)

Plan, Commit, Execute Protocol in Multi-agent Systems

Petr Kadera and Pavel Tichy

Rockwell Automation s.r.o., Pekarska 10a, 15500 Prague 5, Czech Republic
{pkadera,ptichy}@ra.rockwell.com

Abstract. In this paper we introduce changes made in Plan/Commit/Execute negotiation protocol originally developed for Agent Cooperation System and practically used in Chilled Water System application. This protocol is suitable for systems where the commitment of resources and execution of a plan require separated communication phases. The extension of the protocol with Reject Proposal message that is used for canceling unsatisfactory proposals is the most significant change. We also describe the deployment of the protocol in new version of Multi-agent System for Manufacturing.

Keywords: Agents, Distributed Control, Cooperation.

1 Introduction

Various communication protocols are used to standardize communication between agents in the area of Multi-agent Systems (MAS), e.g., voting, negotiation, and auctions [14]. This standardization is useful not only for possibility to easily choose the best protocol for each specific situation, but also for observation purposes and for detection of correct communication behavior of agents.

Contract-Net Protocol (CNP) was firstly proposed by Smith in 1980 [13]. The protocol describes a negotiation in a group of agents and it belongs to auction protocols with characterization of a one-sided, first-price, sealed-bid auction [14]. Any member of the group can temporarily become a manager, who initiates a negotiation process and contacts other agents. When a contacted agent realizes, that he cannot satisfy the order by himself only, he becomes also an initiator of another communication to distribute subtasks among other agents. The manager chooses the best one from the obtained proposals and delegates the contract to the chosen agent.

However, the CNP is one of the most popular research frameworks in the field of distributed systems, it has some disadvantages. One of the most significant is the communication explosion in systems with high branching factor. The problem has been tried to be solved by heuristic in selecting agents that are contacted. This approach was introduced, for example, in [1] or with Branch-and-Bound optimizing [2]. Time delay is another issue that is the CNP facing in real world tasks. A possible solution was introduced in [3]. Proposal of two phase commit protocol is introduced in [10].

The importance of commitment in agent negotiation led to proposal of Competitive CNP [4]. The extension of this protocol is valuable especially in Competitive Multi-Agent Environments. Nevertheless, the Competitive CNP does not contain equivalent execution phase in the protocol that is going to be introduced in this paper and also does not enable to search the complete state space, since the resource allocation is supposed to be done already at the beginning of the negotiation and therefore an agent cannot participate in more concurrent planning operations or in more parts of the same plan.

Proposed improvement of the standard CNP does not tend to reduce communication load, but is focused at definition of the phases of the communication process to enhance potential of the protocol. The communication is separated in three phases – Plan, Commit, and Execute. The goal of the Plan phase is the search for the optimal solution. The responders do not allocate their resources for the initiators in this phase. It enables each agent to take part in more plans or in more parts of the same plan, even if the agent would not have enough resources to satisfy requestors of all plans. The resource allocation comes in the Commit phase, when the chosen responder (it might be also more than one) is requested to reserve its resources for the initiator. The Execute phase is directly connected to output actions usually connected to real hardware or actuators and that is why it can be activated only in case when the Commit phase of all sub-plans has succeeded.

The paper is organized as follows. Section 2 describes communication protocols used in ACS where we focus on Plan/Commit/Execute protocol and its improvements. Section 3 presents real MAS applications that adopt described protocols.

2 ACS Communication Protocols

Each agent in MAS that is designed to interact with other agents usually provides a set of capabilities. When an agent is unavailable for some reason, another agent with the same capability can theoretically compensate for this loss. This flexibility is achieved by dynamical discovery of capabilities in real-time, i.e., the MAS is no longer using unavailable agents and it immediately utilizes new agents. This is achieved, for example, by Directory Facilitator agents [8] in a FIPA compliant system [9].

In Agent Cooperative System ACS [11], each agent is associated with capabilities and each capability is associated with a specific set of operations. By registering with a certain capability, an agent supports all defined operations for that capability and thus can accept and process any request message referring to one of these operations.

Any of the following protocols can be used by an agent to communicate with another agent by sending a set of messages targeted at selected capability/operation.

2.1 Description of Protocols

There are four communication protocols available in ACS system.

- *Inform* only protocol
- *Request/Inform* protocol
- *Request/Inform Plan/Commit/Execute* protocol
- *Request/Inform Plan/Execute* protocol

Inform only protocol is used when an agent wants to send information to another agent without requirement of receiving any information back (e.g., without confirmation), i.e., one way communication only.

Request/Inform protocol means two-way communication and it consist of request message and inform message that is sent back to sender.

Request/Inform Plan/Commit/Execute (PCE) protocol and Request/Inform Plan/Execute (PE) protocol consist of several phases of communication sent between two agents. PCE uses three and PE two phases. Each phase uses Request/Inform protocol internally, e.g., plan phase consists of request plan and inform plan messages.

Plan phase is used for constructing a plan to fulfill some capability/operation request. An agent can either plan to satisfy this request locally and/or it can contact other agents and subcontract them. If a requesting agent needs only one or subset of contacted agents to participate in an execution then the contacted agent usually publish a cost in this plan phase and the requesting agent can make local optimization in this case.

After an agent successfully processed the planning phase, there is an option to use the commit phase to make sure that all resources are committed for this plan. This phase should be used only when there is expectation or possibility that some conflict of resources might occur. All agents that have been selected as part of the plan are contacted. The contacted agents reply with the inform commit message to inform the requester about the successful allocation of resources to carry out the plan. In case the commit phase fails, request to uncommit is sent to all agents who have been already sent commit message.

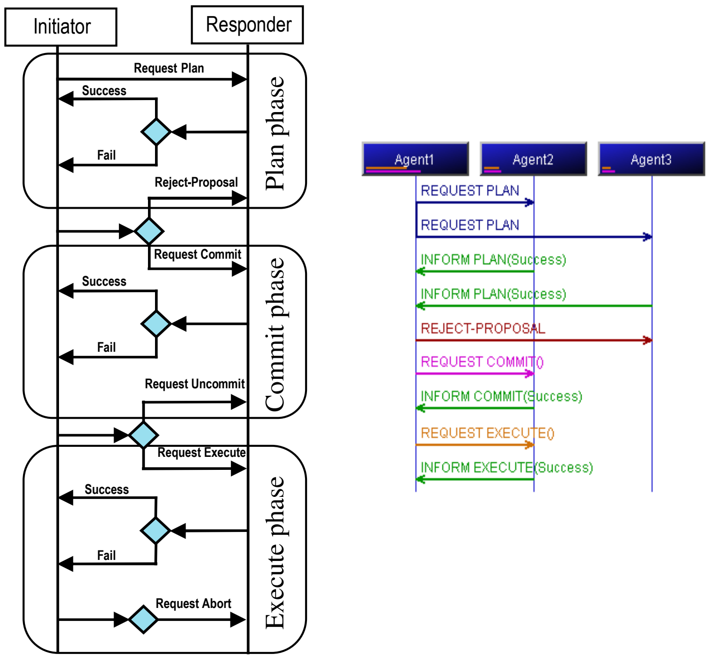


Fig. 1. UML sequential diagram of PCE protocol on the left and PCE protocol example on the right

After an agent successfully processed the planning or possibly also the commit phase, the execution phase occurs. The agents execute their local plans and reply with inform execute messages that contain information about plan execution success or failure status. In case the execution phase fails, request to abort is sent to all agents who have been already sent execute message. The graphical expression of the protocol is shown in Fig. 1. on the left side and the example of one message exchange using this protocol is on the right side.

2.2 PCE/PE Protocol Improvements

Both PCE and PE protocols, described in the previous section, were designed to achieve plan, alternatively commit, and execute phases with minimum messages passed between agents. This is important in the case when the cost of communication (in a form of computation power, communication bandwidth, etc.) is very high. The main drawback of this design is that an agent that responded to plan or commit request have to remember communication context even when this agent was not selected for further commit or execute, because the protocol is missing reject type of message.

For example, agents B and C are contacted by agent A with request for plan. Both B and C agents subcontract agent D and they both remember this with plan details. Both B and C agents inform agent A about successful plan and both provide a cost that is associated. Agent A selects the cheapest one, for example agent B, and it sends request commit message to agent B. At this point agent C still holds information about the plan (that includes, for example, knowledge about subcontracting agent D). Since there is no message that would inform agent C that somebody else has been selected, agent C has to remember this plan for some amount of time and/or until there is no memory left to hold information about plans.

By sending a reject type of message to agents that were not selected the agent have to remember the information about plans only until the reject message arrives. The cost of this improvement is in one extra message per rejected agent (see Fig. 1.). By making this possibility to respond with reject optional, the designer of MAS has a possibility to choose the best approach in given environment based on the balance between communication load and computation cost. In summary, the main disadvantages are the increase of the communication load and the computational power associated with this communication, i.e., message preparation and processing.

The ACS system has been changed accordingly. A collection of plan concentration methods, responsible to select agent(s) that will be considered for further participation, has been improved to automatically send reject message to agent(s) that was(were) not selected.

3 Applications of ACS Communication Protocols

The ACS communication protocols have been already utilized in two projects in Rockwell Automation Research Center. The first one – Chilled Water System (CWS) [5] – solves the problem of distributed control of water flow in a CWS to improve robustness, reliability, and efficiency. The second project is the semantic extension of Multi-agent System for Manufacturing (MAST).

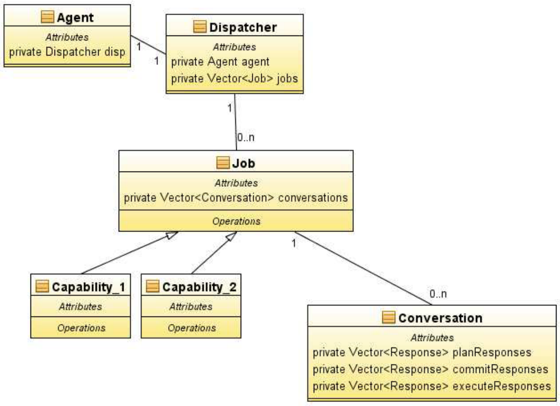


Fig. 2. Class diagram of agent internal structure that uses PCE protocol

The implementation of the protocol is similar in both applications and is illustrated by the class diagram (see Fig. 2.). Each agent is equipped with a dispatcher that is responsible for creating and handling jobs associated with the agent. The Agent and Dispatcher classes are abstract and are extended by every particular user agent and dispatcher (defined for each agent according its capabilities). The job is created, when somebody requests a capability (all extend class Job) of the agent. The job is used mainly to achieve context of conversations since there can be multiple conversations processed in parallel, for example, when an agent has two capabilities and both of them are used at the same time or when one capability is used by two requestors simultaneously.

The dispatcher goes in cycle through the vector of active jobs and performs one step at a time on each of them. The communication with other agents is handled by instances of class Conversation that are kept in vector contained in each job. The conversation object handles responses referring to the job separately for each of the PCE phases.

3.1 Chilled Water System Application

The CWS Application (see Fig. 3.) is a multi-agent project for control of a simulated shipboard chilled water system implemented in Java language. The multi-agent approach provides robustness, flexibility, and scalability. Elements of the CWS are represented by three types of agents. Agents of type “Chiller” provide cold water, agents “Service” require cold water, and agents “Valve” connect segments of water piping system. Agents take actions to dynamically create cooling paths, which enables agents “Service” to stay close to their required temperature. Ability to isolate a leakage, when it occurs, to keep other parts of the system healthy, is another feature of the system. The graphical user interface of the application allows a user to watch running processes in the system as well as edit the system configuration.

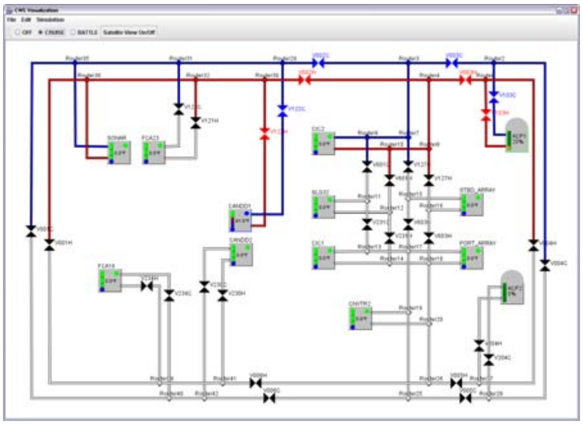


Fig. 3. Chilled Water System simulation and visualization example

The basic feature of the CWS is to provide cooling from chillers to services. This task is in our application reduced to creating paths that connect these two types of components. A service has to be connected to get cooling to a chiller with a path for cold water transportation to the service and with another path for hot water transportation from the service back to the chiller. Hot water needs to be returned to the same chiller to avoid lack or overflow of water in its reservoir with limited capacity. Paths are composed of pipe sections connected by opening the relevant valves. Every valve has its own opening price as well as each chiller has its price for switching on.

Let us take a look at the planning used in this application from general point of view. The search is processed concurrently for cold and hot water paths. The distribution of the planning process is based on storing partial planning results. Each agent involved in the process modifies parameters in the communication act. Parameter "Path" is extended with name of the agent and Parameter "Cost" is increased with cost of the valve and the planning continues with next agent as is introduced below.

Despite the fact that the initiator of cooling process is always a service agent, the path search starts from each chiller in parallel. This approach enables to break the process of planning at the beginning, in case there is not enough free cooling capacity that requests the service. The chiller modifies the planning parameter "Path" (fills in the agent's name) and "Cost" (fills in the chiller's fee for providing cooling – in this application there is considered fee only for the case, the cooling requires switching on the chiller). Among chiller's parameters are also two lists ("Cold Valves" and "Hot Valves") that contain names of neighbouring valves.

The example of planning process is shown in Fig. 4. The service CIC2 (marked with number 1) sends the RequestPlan message to both chillers (2 and 3) that send RequestPlan messages to their neighboring valves. The planning process continues until a valve that is directly connected to the service is reached. Then the message

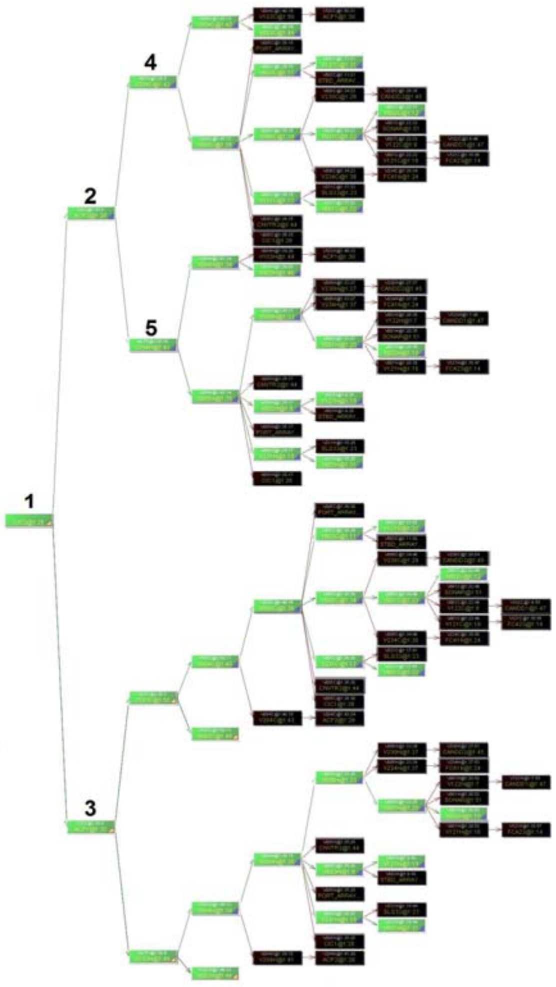


Fig. 4. CWS path planning communication workflow example

Inform Plan (Success) is sent and propagated through the chain of agents in the opposite direction. If the service is not reached then Inform Plan (Fail) is sent. When initiator of the planning receives the result, he selects the agent with the lowest cost and sends him request commit. The initiator sends message Reject-Proposal to the others. If the commit phase succeeds in all the chosen agents, resulted Inform Commit (Success) is sent to the initiator who continues by sending Request Execute and expects answer Inform Execute (Success). In case the commit phase fails, all agents who have been already sent Request Commit are sent Request Uncommit. Similarly, in the case of failure in commit phase the Request Abort is sent to all agents who have been sent Request Execute message.

3.2 MAST

MAST [12] was originally designed to demonstrate the advantages of multi-agent approach in field of manufacturing. The application controls, visualizes (see Fig. 5.), and simulates material transportation in a production line. The line might consist from elements such as follows:

- Conveyor Belts (marked with number 1 in Fig. 5),
- Automatic Guided Vehicles,
- Docking Stations (2),
- Diverters (3),
- Robots (4), and
- Storages (5).

The last version of MAST has been extended with communication protocol based on the protocol that was developed for ACS platform, however, MAST is based on different multi-agent platform – JADE [14]. The behavior of agents is implemented in similar way as in ACS platform. There are also capabilities that are assigned to particular agent classes and enable them to react to incoming requests.

The main agent types used in this application are as follows:

- *Order* agent – responsible for accepting an order and creating a product agent for each ordered item,
- *Product* agent – requires production plan from Production Plan agent and processes successive execution of steps from the plan,

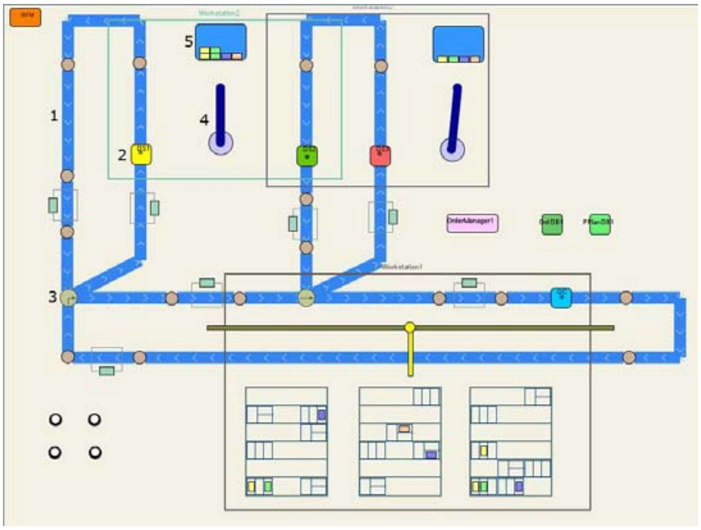


Fig. 5. MAST transportation example

- *Production Plan* agent – storage of production plans for all possible products,
- *Transporter* agent – provides transportation for product agent using transportation equipment (Conveyor belts or AGVs), and
- *Workstation* agent – encapsulates capabilities of elementary production line equipment (e.g., robots, drills, etc.) to provide complex capabilities for product agents.

The PCE protocol is used for dynamic and distributed creation of plans for fulfilling the goal of product making. The resultant plan contains contracts with transportation agents (Conveyor belts or AGVs) and production acts (e.g., adding a part to a product.

4 Conclusions

The main ideas presented in this paper are overview of CNP extensions, introduction of Plan/Commit/Execute Protocol, description of its improvements, and examples of practical usage of the protocol. The usage of Plan/Commit/Execute Protocol enables separation of planning process from resource allocation to enable search through the complete state space without influence by the allocation order. The proposed improvements are based on the utilization of new messages – Request Uncommit and Request Abort. The practical usage of this protocol is demonstrated on CWS application and the last version of MAST.

References

1. Ohko, T., Hiraki, K., Anzai, Y.: Addressee Learning and Message Interception for Communication Load Reduction in Multi-robot Environments. Distributed Artificial Intelligence Meets Machine Learning, pp. 242–258. Springer, Berlin (1997)
2. Chun, A.H.W., Wong, R.Y.M.: Optimizing Agent-Based Negotiations with Branch-and-Bound. In: Proc. 6th International Computer Science Conference, Hong Kong, China, pp. 235–243 (2001)
3. Qiaoyun, L., Jiandong, L., Dawei, D., Lishan, K.: An extension of contract net protocol with real time constraints, pp. 156–162. Wuhan University Journal of Natural Sciences, Wuhan (1996)
4. Vokřínek, J., Bíba, J., Hodík, J., Vybíhal, J., Pěchouček, M.: Competitive Contract Net Protocol. In: Proc. 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, pp. 20–26 (2007)
5. Tichý, P., Šlechta, P., Maturana, F.P., Staron, R.J., Hall, K.H., Mařík, V., Discenzo, F.M.: Multi-agent Technology for Robust Control of Shipboard Chilled Water System. In: Proc. International Federation of Automatic Control (IFAC) Conference on Control Applications in Marine Systems, session TA2, Ancona, Italy (2004)
6. Tichý, P., Šlechta, P., Maturana, F.P., Balasubramanian, S.: Industrial MAS for Planning and Control. In: Mařík, V., Štěpánková, O., Krautwurmová, H., Luck, M. (eds.) ACAI 2001, EASSS 2001, AEMAS 2001, and HoloMAS 2001. LNCS, vol. 2322, pp. 280–295. Springer, Heidelberg (2002)

7. Vrba, P., Mařík, V.: Simulation in agent-based control systems: MAST case study. International Journal of Manufacturing Technology and Management 8(1/2/3), 175–187 (2006)
8. Tichý, P.: Middle-agents Organized in Fault Tolerant and Fixed Scalable Structure. Computing and Informatics 22, 597–622 (2003)
9. FIPA: The Foundation for Intelligent Physical Agents, Geneva, Switzerland (1997), `http://www.fipa.org`
10. Nimis, J., Lockemann, P.C.: Robust Multi-Agent-Systems. The Transactional Conversation Approach. In: Proc. of SASEMAS, New York (2004)
11. Tichý, P., Šlechta, P., Staron, R.J., Maturana, F.P., Hall, K.H.: Multiagent Technology for Fault Tolerance and Flexible Control. IEEE Transactions on Systems, Man, and Cybernetics 36, 700–705 (2006)
12. Vrba, P.: Simulation in Agent-based Control Systems: MAST Case Study. International Journal of Manufacturing Technology and Management 8(1/2/3), 175–187 (2006)
13. Smith, R.: The Contract Net Protocol: High-level Communication and Control in Distributed Problem Solver. IEEE Transactions on Computers 29(12), 1104–1113 (1980)
14. Jennings, N.R., Bussmann, S., Wooldrige, M.: Multiagent Systems for Manufacturing Control. Springer, Berlin (2004)
15. JADE agent platform (2004), `http://jade.cselt.it`

Distributed Sensing and Control Architecture for Automotive Factory Automation

Ningxu Cai and Robert W. Brennan

Schulich School of Engineering, University of Calgary, Calgary, AB, T2N 1N4
Canada, Tel.: 403-220-4192
rbrennan@ucalgary.ca

Abstract. In this paper we propose an architecture for distributed intelligent sensing and control (DISC) for automotive factory automation. The architecture is based on a platform-based design approach that breaks the sensing and control problem into increasingly higher levels of abstraction from physical hardware to intelligent sensing and control. The focus of this paper is on the middleware layer that serves as an interface between an upper, agent-based control level and the wireless sensor network. This layer takes advantage the IEC 61499 model for distributed process measurement and control, and in particular, exploits its distributed, modular structure and its close match to wireless sensor systems.

Keywords: industrial automation, wireless sensor networks, IEC 61499.

1 Introduction

While wireless sensor network (WSN) technology has been very successful in military and defence application during the past two decades, it is also rapidly becoming a viable solution for employment at the lowest level of factory automation systems (Pellegrini et al., 2006). Challenges for the applications of distributed sensor systems in manufacturing environments include harsh, uncertain, dynamic shop conditions, as well as integration with new control software approaches to realize the flexibility and responsiveness of the whole system.

In this paper, a WSN architecture is proposed that is intended to exploit the potential of distributed sensor systems to address the challenges faced by today's automotive manufactures: i.e., the combination of increasing stringent customer requirements (e.g. high quality, customizable, low-cost products that can be delivered quickly) and inherent manufacturing system complexity (i.e. these system are by nature, distributed concurrent and stochastic). The primary objective of this work is to develop an efficient shop floor sensing and control system to support real time decision making in automotive factory automation systems.

This project is part of the Canadian AUTO21 Networks of Centres of Excellence: a nation-wide, interdisciplinary research effort that is focused on automotive research. Our contribution to AUTO21 is a collaborative venture between researchers at the University of Windsor, the University of Calgary, and the University of Western

Ontario and spans the disciplines of micro-sensor design, embedded real-time control, and agent-based systems.

This paper begins with background on the wireless sensor networks in factory automation. Next we describe the overall architecture for our proposed distributed intelligent sensing and control (DISC) system in Section 3. This architecture follows a platform-based design approach that is intended to support collaborative work on each level of control: the physical layer (University of Windsor), the middleware (University of Calgary), and the application layer (University of Western Ontario). For the remainder of the paper we focus on the work at the University of Calgary with an overview of the challenges associated with wireless sensing platforms in Section 4 and a description of the implementation plan in Section 5. The paper concludes with a summary and discussion of future work in Section 6.

2 Background

Conventional factory automation systems rely on a set of hard-wired sensors that are either directly linked to controller I/O devices or are part of a sensor network. These systems are expensive to install and difficult to maintain, and limit future expansion due to an inflexible overall layout.

Wireless integrated network sensors offer a low-cost solution that can cover the entire enterprise. These sensors contain multiple heterogeneous on-board sensors that are networked through wireless links and are deployable in large numbers. This system-wide deployment of sensing devices is referred to as distributed sensing, and the whole infrastructure is called a distributed sensor system.

WSN technology has resulted in a data-rich environment with both temporally and spatially dense information that provides unprecedented opportunities for product quality and productivity improvement.

However, applications of WSNs in manufacturing environments are very different from other applications: e.g., military and defence sensor networks and wireless environmental monitoring systems, where sensors are usually deployed in the open fields or in the ocean. In a typical manufacturing system, various noise and vibration sources as well as diverse electrical and electronic systems, all have significant influence on the performance and efficiency of wireless sensor systems. As a result, smart sensors and communication protocols that have been successfully deployed in other applications will not necessarily work in the harsh, uncertain, dynamic environments typical to automotive manufacturing. Therefore, significant research efforts are required in this area. Additionally, WSN hardware technology alone is not sufficient to address the challenges faced by today's automotive manufacturers: this technology must be integrated with new control software approaches to realize systems that are flexible and responsive to the ever-changing manufacturing environment.

3 System Architecture

This paper focuses on the first steps towards the development of efficient shop floor wireless sensing technologies to support real time decision making in automotive factory automation systems: i.e., the system architecture. To achieve this goal, a platform based design approach for WSNs proposed by Bonivento and Cailoni (2006) is adopted and modified for system design and functional analysis.

Platform based design is a meet-in-the-middle approach where the top-down refinement of a design specification meets with bottom-up characterizations of possible alternative implementations, which encourages each team member to contribute to the project from their own perspective. In the end, the design space exploration is performed based on estimates of the performance of candidate solutions so that the overall design process is considerably speed up as re-designs are avoided and design re-use is favoured. Based on this, the integration of different task modules within the whole project can be accomplished efficiently.

Our architecture for distributed intelligent sensing and control for automotive factory automation is illustrated in Figure 1.

The WSN platform consists of three abstracted layers: the Sensor Network Service Platform (SNSP), the Sensor Network Ad-hoc Protocol Platform (SNAPP), and the Sensor Network Implementation Platform (SNIP). At the physical device level, the investigation of low power wireless sensors for installation on rotating parts within typical automotive manufacturing environment forms another key research area of this project. The overall system requirements, such as throughput, latency, reliability, security, adaptability, affordability and energy consumption, will be satisfied through the functional integration of different task modules. Descriptions of the four task modules at different levels of the proposed distributed intelligent sensing and control (DISC) system are as follows.

The Sensor Network Service Platform (SNSP) layer is the upper application interface for our WSN system. It defines services available to the user, which include functionalities of sensing, control and actuation. The SNSP can be thought of as the interface to the DISC as it serves as a purely functional description for the system: i.e., the detailed application specific network implementation is not dealt at this level.

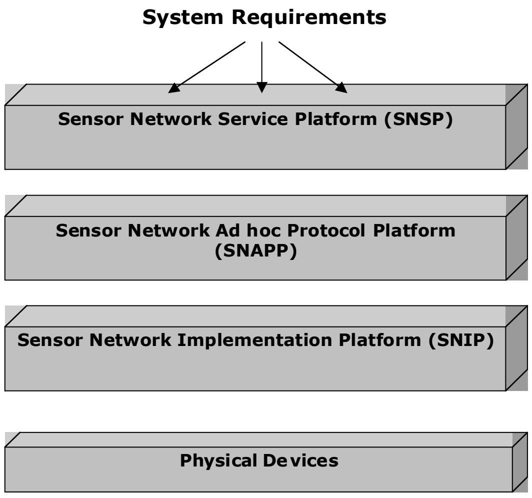

System Requirements
Sensor Network Service Platform (SNSP)
Sensor Network Ad hoc Protocol Platform (SNAPP)
Sensor Network Implementation Platform (SNIP)
Physical Devices

Fig. 1. Distributed intelligent sensing and control system architecture

The SNSP decomposes and refines the interaction among controllers by providing services, such as query, command, timing/synchronization, location, and concept repository.

The Sensor Network Ad-hoc Protocol Platform (SNAPP) defines the software architecture for the physically distributed sensor nodes and maps the functional specification from SNSP level onto sensor nodes. For the proposed DISC system, intelligent software agents will play an important role in this area. In particular, agent clustering and mediation approaches (Shen and Norrie, 1999) will be used to address topology changes of the ad hoc WSN; agent-based distributed decision making mechanisms will be developed and applied for collaborative signal and information processing. An example of this approach is described in (Shen et al., 2005).

The SNAPP will be supported by a library of MAC and routing protocols. These protocols are "parameterized protocols", and the parameters of system working points can be obtained as the solution of a constrained optimization problem, where the constraints are derived from system requirements and characteristics of physical nodes.

The Sensor Network Implementation Platform (SNIP) functions as a middleware to build up a network of interconnected physical nodes that implement the logical function of the application. At this level, both wireless communication standards for networks and sensors need to be considered. SNIP acts as a bottom layer of the software to directly interface with physical sensors. To achieve real-time sensing and control, reconfigurablility, as one of the key requirements of distributed sensing networks, will be the research focus for this task module. The challenges associated with the SNIP layer and its preliminary architecture will be described in more detail in sections 4 and 5 respectively.

4 Challenges Associated with Distributed Sensing and Control Platforms

According to the system structure described in Section 3, the tasks of three WSN platform modules of our distributed sensing and control system focus mainly on software/middleware. All three of these layers are closely interconnected. It is intended that the integration of these three layers will provide a new control approach to realize systems that are flexible and responsible to the ever-changing manufacturing environment.

In today's market, nearly all wireless sensor vendors' software can provide some level of node configuration, offer information on the state of each node, display real-time wireless sensor data on a graph for monitoring purposes, and even provide an interface that allows the user to set up basic data logging, making it possible to export data to a spreadsheet for offline analysis (Hobbs, 2006). Although these software tools are fairly intuitive, they typically have fixed capabilities and lack several key features to support real-time decision making in automotive factory automation systems. As a result, novel programming paradigms and new technologies are required for our distributed sensing and control platforms.

More specifically, the proposed Distributed Intelligent Sensing and Control (DISC) platform for factory automation will focus on three key challenges:

1. self-organization,
2. embedded node intelligence, and
3. low-power operation.

4.1 Self-organization

DISC self-organization can be thought of as dynamic reconfiguration (Brennan et al., 2008) of the system nodes. In other words, the DISC system must be capable of accommodating changes to sensor node configuration automatically, as the system operates. Examples of sensor node reconfiguration may include the addition of new sensor node, reassignment of existing sensor nodes, or the removal of obsolete sensor nodes.

Ad hoc wireless sensor networks rely on a fully distributed architecture where there is no central controller (Stankovic, 2003). All operations (access control to the radio medium, routing, etc.) are data centric and application-oriented. Explicit node addressing via ID or IP is unfavourable since random node distribution and mobility impedes assignment of node addresses to the position of a measurement. As a paradigm shift, the network topology needs to be constructed in real-time and updated in a self-configuring fashion periodically as sensors fail or move and new sensors are deployed (Deb et al., 2001).

4.2 Embedded Node Intelligence

Given the processing limitations of WSN technology, one must be realistic about the DISC system's ability to support embedded node intelligence. More specifically, although it is unrealistic to expect extant wireless sensor nodes to support software agents with high-level reasoning capabilities, it is reasonable to exploit the node's hardware and software platform to enhance the node's intelligence in areas such as signal processing, alarm monitoring, and node diagnostics.

Current wireless sensors are passive nodes that simply pass the data they are hard-coded to provide back to the user. Few have built-in intelligence for data analysis or automated power management. For example, a node might be embedded in a large machine to monitor vibration levels. Although it can acquire a large amount of raw data, it may need to send only pass/fail information to the host, indicating whether the machine is within acceptable limits (Hobbs, 2006). Intelligent nodes are essential for the higher level monitoring and control, and can further improve the intelligence of the whole system.

Embedding intelligence into sensor nodes has the advantage of redistributing some of the overall processing load from the upper-level controllers to the sensor system. More importantly, for the purposes of our DISC system, it supports the platform-based design approach to dynamic reconfiguration and ultimately, intelligent reconfiguration at the higher agent levels. In other words, by shielding the upper layers from the hardware implementation details, the lower-level SNIP layer provides a common interface to the WSN. For example, at the SNSP level, one is not concerned whether a new sensor is a Type X or a Type Y vibration sensor (requiring configuration X or configuration Y respectively): only that the new sensor is a vibration sensor.

4.3 Low Power Operation

Closely related to the processing limitations of WSN technology is the low power operation limitation of these systems. Normally sensor networks run on small batteries and often need to operate for a long time: power conservation is a key issue at all layers in sensor networks. Besides the effort put at the physical device level, recent studies have shown that radio communication is the dominant consumer of energy in sensor networks (Hill et al., 2000). As a result, implementing a low-power task scheduling, communication protocols are all important for energy conservation.

For the proposed DISC system, power consumption will be a key constraint of our system design. Although increased node intelligence will likely result in higher power consumption at the node (i.e., increased processor load), this should be offset by decreased node communication requirements. In particular, node-level signal processing will reduce the amount of raw data that will be transmitted from the sensor node to the upper, controller levels of the DISC system.

The three key challenges described in this section figure prominently in the design of a DISC system. Of course, these design considerations must be addressed in addition to the typical challenges associated with WSN implementations. In particular, security, predictability, real time performance, and integration with the rest of the enterprise need to be taken in to consideration as well for wireless sensor network platform design.

5 The Sensor Network Implementation Platform

In this section, we look more closely at the implementation of the SNIP (sensor network implementation platform) level of the DISC. As noted previously, the SNIP acts as a middleware between the agent-based software at the SNAPP level and the physical sensor nodes. In other words, the SNIP provides an interface between upper level software agents and lower level physical devices that abstracts the implementation details at the device level to support intelligent control at the level of the agent system. As a result, embedded intelligence and interface design are key concerns at the SNIP level.

The SNIP level's tight link to physical devices and the DISC system's requirement for a distributed software model are, consequently, a very good match to the IEC 61499 model for distributed process measurement and control (IEC, 2005). As well, recent work on real-time reconfiguration services for function block based systems (Zoitl, 2006) should support the ultimate software/hardware implementation at this level.

Some early research efforts utilizing function blocks include reconfiguration of real-time distributed systems (Brennan et al., 2002), holonic control (Wang et al., 2001), function block oriented engineering support systems (Thramboulidis and Tranoris, 2001), Web-based engineering and maintenance of distributed control systems (Schwab et al., 2005), and OOONEIDA initiative where function block can serve as a cornerstone for further development of the automation object concept due to its characteristics of portability and reusability (Vyatkin and Christensen, 2005).

5.1 WSN Sensor Nodes

Before we look at the relationship between the SNIP and the IEC 61499 model, we will first look more closely at the basic structure of a sensor node. As illustrated in Figure 2, a physical sensor node is a collection of physical resources, such as clocks

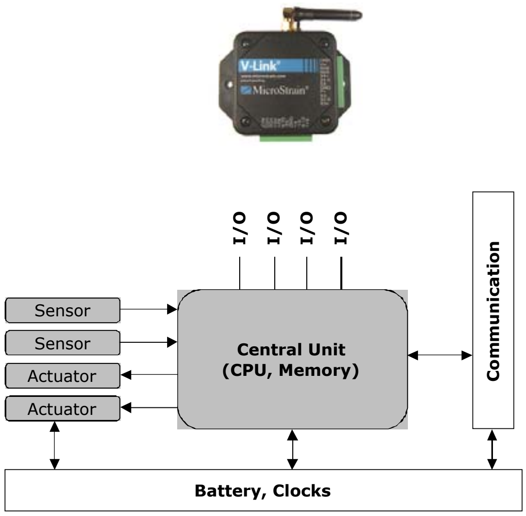


Fig. 2. Basic structure of a sensor node

and battery resources; processing units, memory, communication and I/O devices; sensor and actuator devices.

This figure also shows an example of a typical commercial sensor node: i.e., a V-Link® Wireless Voltage Node (MicroStrain, 2008). This particular sensor node utilizes the IEEE 802.15.4 open communication standard and supports simultaneous streaming from multiple nodes. Its onboard memory stores up to 1 x 106 measurements, and its 3-volt sensor excitation supports most analogue sensors.

5.2 The DISC and IEC 61499

As noted, given the IEC 61499 model's focus on embedded and distributed control (Vyatkin, 2007; Zoitl, 2006), it provides a natural mapping to our proposed DISC system. More specifically, there is a direct mapping between IEC 61499 devices and WSN sensor nodes as illustrated in Figure 3.

This figure shows the bottom three layers of the DISC architecture illustrated in Figure 1. At the sensor node implementation platform (SNIP), sensor nodes are represented by IEC 61499 devices that are linked by a WSN. Embedded node intelligence is supported by IEC 61499 function block applications (FBA).

The key to the mapping lies in the IEC 61499 modular software model. As noted previously, the main SNIP design issues are interface design and embedded intelligence. As a result, the SNIP will utilize a library of device level services – in the form of IEC 61499 service interface function blocks (SIFB) – to support WSN communication and the process interface. Given the typical sensor node structure (illustrated in Figure 2), the process interface will be at the level of the node's central unit: i.e., the sensor interface will be handled by the node's underlying hardware and software.

Sensor node embedded intelligence will be a combination of service interfaces (i.e., where some signal processing and diagnostic services are already provided) and custom function block applications. In the latter case, basic and composite function

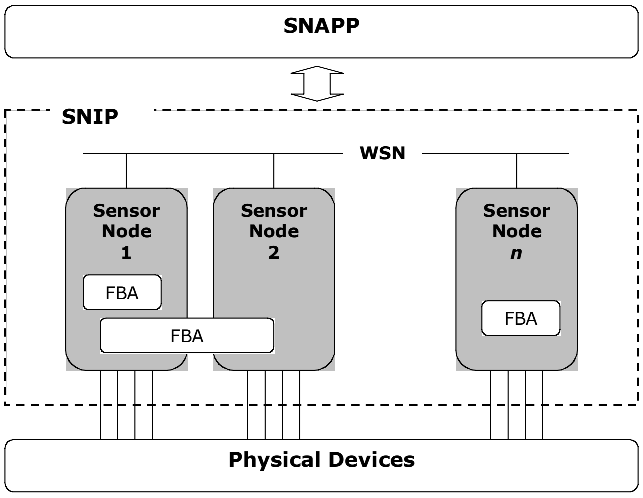


Fig. 3. Mapping between the IEC 61499 and the DISC system

block models can be exploited to develop custom function block applications at the SNIP level.

Given that "embedded intelligence" at this level of the DISC means a masking of the sensor node's implementation details from the agent-level (i.e., the SNAPP) and basic diagnostic services, the function block applications will map to individual sensor nodes (i.e., they will not be distributed across nodes). However, it is conceivable that higher levels on embedded intelligence may be possible at the SNIP that would require distributed function block applications as illustrated in Figure 3. For example, simple fault monitoring an recovery could be dealt with at the SNIP without the need to defer to the agent level. However, a key constraint that will need to be considered will be the power drain that would result from increased inter-node communication.

6 Summary

This paper provides a summary of the basic architecture for a distributed intelligent sensing and control (DISC) system for automotive factory automation. At the University of Calgary, we are focusing on the main issues of sensor network implementation at the middleware level and propose the IEC 61499 model to support sensor node embedded intelligence and distributed sensing and control.

Intelligence at the lowest levels of the DISC is intended to enhance sensor node functionality, performance, reliability, and facilitate sensor node integrated with the entire system (e.g., local analysis and node aggregation). Local data analysis means only parametric data needs to be passed to the upper level system. For example, a node might be embedded in a large machine to monitor vibration levels. Although it can acquire a large amount of raw data, it may need to send only pass/fail information to the host, indicating whether the machine is within acceptable limits (Hobbs, 2006).

While the SNIP layer focuses on the functionality of the application, the SNAPP layer provides the services of sensor discovery and collaborative signal processing. To integrate these two layers, security and real-time issues are involved. Communication interfaces at different levels of function block network build up an information channel between the SNIP and SNAPP levels. The analysis capability and programmability of IEC 61499 function blocks make system simulation, execution and monitoring at higher levels possible, and have the potential to further guarantee the security and real-time requirements of the overall system.

Acknowledgements

The authors wish to thank the AUTO21 Networks of Centres of Excellence and the Natural Sciences and Engineering Research Council of Canada for their generous support of this research. We also thank our research partners, Dr. J. Wu and Dr. C. Chen (University of Windsor) and Dr. W. Shen (University of Western Ontario) for their input and discussions on various aspects of this work.

References

1. Bonivento, A., Carloni, L.P.: Platform based design for wireless sensor networks. Mobile Networks and Applications 11(4), 469–485 (2006)
2. Brennan, R.W., Vrba, P., Tichy, P., Zoitl, A., Sünder, C., Strasser, T., Marik, V.: Developments in dynamic and intelligent reconfiguration of industrial automation. Computers in Industry 59, 533–547 (2008)
3. Brennan, R.W., Zhang, X., Xu, Y., Norrie, D.H.: A reconfigurable concurrent function block model and its implementation in real-time Java. Integrated Computer-Aided Engineering 9, 263–279 (2002)
4. Deb, B., Bhatnagar, S., Nath, B.: A topology discovery algorithm for sensor networks with applications to network managements. Dept. Comput. Sci., Rutgers Univ., Tech. Rep. DCS-TR-441 (2001)
5. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., Pister, K.S.J.: System architecture directions for network sensors. In: Proc. Int. Conf. Architectural support for Programming Languages and Operating Systems (ASPLOS), Cambridge, MA (2000)
6. Hobbs, K.: Wireless sensor networking software - the next generation (2006), `http://www.sensorsmag.com/sensors/Technology+Tutorials/Networking+&+Communications/Wireless-Sensor-Networking-Software-The-Next-Gener/ArticleStandard/Article/detail/314744`
7. International Electrotechnical Commission, Function Blocks – Part 1: Architecture, International Electrotechnical Commission, Geneva, Switzerland, Technical Report IEC 61499-1 (2005)
8. MicroStrain wireless sensor (2008), `http://www.microstrain.com/v-link.aspx`
9. Pellegrini, F.D., Miorandi, D., Vitturi, S., Zanella, A.: On the use of wireless networks at low level of factory automation systems. IEEE Transactions on Industrial Informatics 2(2), 129–143 (2006)

10. Schwab, C., Tangermann, M., Ferrarini, L.: Web based methodology for engineering and maintenance of distributed control systems: the TORERO approach. In: Proceedings of the 3rd IEEE International Conference on Industrial Informatics (2005)
11. Shen, W., Norrie, D.H.: Agent-based systems for intelligent manufacturing: a state-of-the-art survey. Knowledge Information Systems 1, 129–156 (1999)
12. Shen, W., Lang, S., Wang, L.: iShopFloor: an internet-enabled agent-based intelligent shop. IEEE Transactions on Systems, Man, and Cybernetics, Part C 35(3), 371–381 (2005)
13. Stankovic, J.A., Abdelzaher, T.F., Lu, C., Sha, L., Hou, J.C.: Real-time communication and coordination in embedded sensor networks. Proceedings of the IEEE 91(7), 1002–1022 (2003)
14. Thramboulidis, K., Tranoris, C.: An architecture for the development of function block oriented engineering support systems. In: Proceedings of IEEE International Conference on Computational Intelligence in Robotics and Automation (2001)
15. Vyatkin, V.V., Christensen, J.H.: OOONEIDA: An open, object-oriented knowledge economy for intelligent industrial automation. IEEE Transaction on Industrial Informatics 1, 4–17 (2005)
16. Vyatkin, V.: IEC 61499 Function Blocks for Embedded and Distributed Control System Design. ISA Press (2007)
17. Wang, L., Brennan, R.W., Balasubramanian, S., Norrie, D.H.: Realizing holonic control with function blocks. Integrated Computer-Aided Engineering 8, 81–93 (2001)
18. Zoitl, A.: Basic Real-Time Reconfiguration Services for Zero Down-Time Automation Systems, Dissertation, Technical University of Vienna (2006)

MAS-Based Cooperative Control for Biotechnological Process-A Case Study

Dariusz Choinski, Mieczyslaw Metzger, and Witold Nocon

Faculty of Automatic Control, Electronics and Computer Science
Silesian University of Technology,
ul. Akademicka 16, 44-100 Gliwice, Poland
{dariusz.choinski,mieczyslaw.metzger,witold.nocon}@polsl.pl

Abstract. The MAS-based control seems to be better suited for manufacturing control because of its discrete event character. Nevertheless, for continuous industrial processes the MAS-based control can be also very attractive. In this paper, a synthesis of the MAS-based control system for the continuous process is presented. A biological reactor is controlled using respirometric approach. In this approach, both standard control loops and additional experiments are performed to obtain measurements not available on-line. Therefore, the system consists of different control agents being able to cooperate or to inhibit each other in order to achieve the appropriate goals. A case study experiments are realized using an experimental wastewater treatment pilot plant.

Keywords: MAS, industrial application of MAS, cooperative control, biotechnological reactor.

1 Introduction

Typical control systems of continuous processes are dominated by classical control systems based on strictly deterministic control algorithms. An example of such a control system is the PID controller. Although, such control architectures may become very complex because of the size of the plant being controlled, the whole control system is in most cases implemented as one coherent system, having a cascade structure. That is, low level control loops receive set point values from the higher level control algorithms. An example of a low level control loop is a simple flow control in a pipe. A higher level control algorithm may for example be responsible for process effectiveness or product quality.

In this paper a slightly different approach is presented and demonstrated for a selected part of a biotechnological plant. Although biotechnological processes are usually considered as continuous processes, a major unpredictable factor exists in such processes, since the object being controlled is a colony of living organisms. The approach presented in this paper is based on a Multi-Agent System (MAS) approach, in which a number of independent software applications (agents) cooperate [1-5]. A number of practical applications of such systems have been developed both in industry and in pilot experimental plants used in research [6-12]. Cooperation in MAS may include agents that use other agents to accomplish a given goal, or periodically prevent other agents from accomplishing theirs goals in order to achieve other tasks required. The MAS approach may for example be used to enhance web-based remote monitoring, experimentation and control [13], [14]. A new parallel processing Producer Distributor Consumer (ppPDC) transmission for MAS distributed control has also been presented [15].

In this paper such an approach is implemented in the field of biological aerobic reactors, controlled and monitored by means of respirometry [16]. The problem under consideration is controlling the aeration process in a biological reactor and at the same time monitoring respiration related activities in the reactor in order to influence biological processes. The need for cooperation of agents arise from the fact, that in order to measure the parameters characterizing respiration related activities in the reactor, the control of dissolved oxygen in the reactor must be suspended. The two parameters that need to be estimated are Oxygen Uptake Rate (OUR) which characterizes the amount of oxygen being taken up by biomass per unit of time [17], [18], and an oxygen mass transfer coefficient kLa that characterizes the ability of aeration device to force oxygen from bubbles into the liquid phase in the reactor. Information about respiration of biomass may than be used by other agents to influence the mass balance of activated sludge in the plant, thus indirectly controlling metabolic processes occurring in living organisms that depend on biomass concentration. One example of using OUR measurement is the optimisation of storage driven denitrification in activated sludge sequencing batch reactors [19]. Additionally, kLa depends on other parameters like mixing regime and mixing power, but also on activated sludge flocs structure and size. Therefore, measuring this parameter and changing hydraulic flows and mixing regimes in the system may in effect influence the kLa itself. A number of publications deal with determination of this parameter [20], [21].

2 Biotechnical Process under Consideration

Experimental studies using the described MAS have been performed on a biotechnological pilot plant designed, developed and operated at the Institute of Automatic

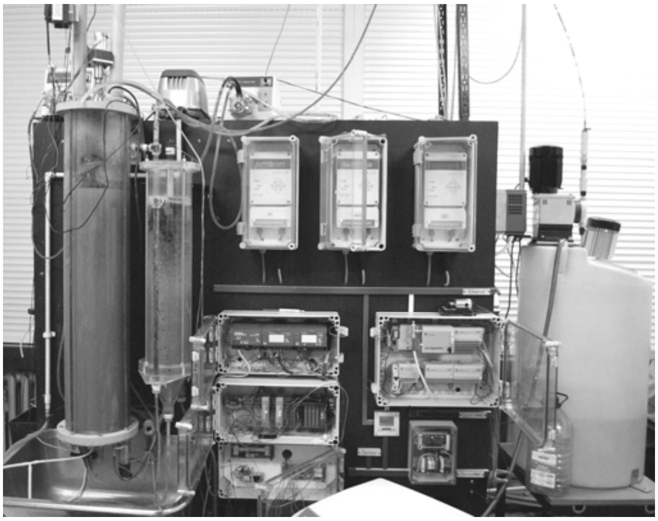


Fig. 1. Schematic layout of the biotechnological pilot-plant under consideration

Control (Fig. 1). The plant, that resembles an activated sludge wastewater treatment system may be operated in different modes: as a classical continuous system (reactor and secondary settler); as a classical sequencing batch reactor; or, as proposed by the authors, a two-tank-SBR in which a sequencing reactor is supplemented with a second tank used for biomass retention when needed. The pilot plant is controlled automatically by a distributed control system enabling two different controller brands to be used for control algorithms implementation, namely, a FlexLogix controller (Rockwell Automation) and FieldPoint controller (National Instruments). Usage of a distributed system designed specifically for intersystem integration is especially useful for MAS implementation.

3 MAS-Based Cooperative Control for Aeration in Biological Reactor

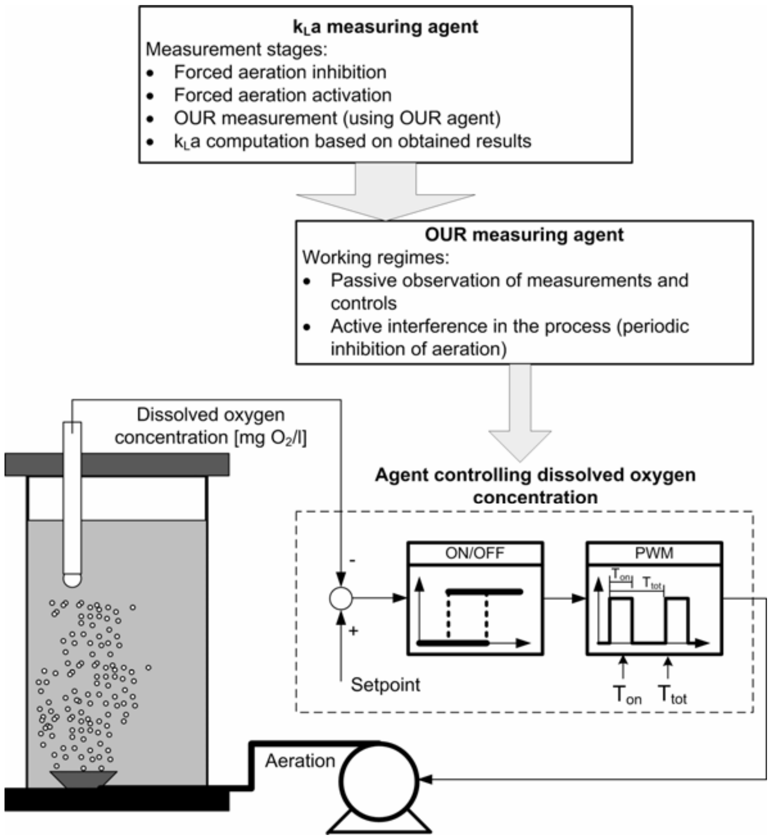
The pilot plant control system contains software modules designed to measure additional parameters that are not measurable in a direct way. Since obtaining those measurements requires an interference into the way the process is being controlled, those software modules have been designed using the agent approach, that is as intelligent independent modules, possessing the capability of periodically taking over control of selected parts of the process.

Parameters measured in an in-direct way are:

- OUR - Oxygen Uptake Rate – a parameter defining a rate of oxygen uptake by the microorganism in the biological reactor. OUR is one of the most important parameters describing the activity of the active biomass.
- k_La – coefficient defining the mass of oxygen being transferred from the gas phase (air pumped by diffusers into the reactor) into the liquid phase (the water with suspended activated sludge flocs).

A classical measurement of oxygen uptake rate (OUR) is carried out by filling a sealed container with a sample of mixed liquor containing wastewater and activated sludge microorganism and measuring dissolved oxygen. Due to the microorganisms oxygen demand caused by the substrate present in wastewater, concentration of dissolved oxygen will decrease with time. The first derivative of dissolved oxygen concentration defines the rate of oxygen uptake by microorganisms.

In case of a working biological reactor, it is neither possible to seal the reactor, nor it is possible to inhibit aeration for a longer period of time due to the possibility of changing the conditions in the reactor. Therefore, the OUR measurement must be realized in connection with the dissolved oxygen control system already implemented for the reactor. This control system is realized by a constant flow pump used for aeration. However, the oxygen demand varies with time, creating a need to change the power of aeration. Since the pump is a constant flow pump, the only way to change gain of such an actuator is to implement Pulse Width Modulation (PWM) control for the dissolved oxygen control. On top of the PWM modulation algorithm, a simple ON/OFF-with-hysteresis algorithm is implemented that switches the aeration when the value of dissolved oxygen drops below the setpoint value (Fig. 2).



k$_L$a measuring agent
Measurement stages:
• Forced aeration inhibition
• Forced aeration activation
• OUR measurement (using OUR agent)
• k$_L$a computation based on obtained results
OUR measuring agent
Working regimes:
• Passive observation of measurements and controls
• Active interference in the process (periodic inhibition of aeration)
Dissolved oxygen concentration [mg O$_2$/l]
Agent controlling dissolved oxygen concentration
ON/OFF
PWM
T$_{on}$
T$_{tot}$
-
+
Setpoint
T$_{on}$ T$_{tot}$
Aeration

Fig. 2. Schematic of the three agents realizing the presented MAS enabling measurement of OUR and k_La values

The agent measuring OUR may work in two different regimes:

- The first regime calls for a passive observation of the available measurements and used controls. In this particular case, the necessary measurement is the dissolved oxygen concentration and the control is the aeration of the pump. The agent observes periods between aerations looking for segments in the dissolved oxygen plot possessing the greatest value of dissolved oxygen decrease. This value is than assumed to be the oxygen uptake rate that is supposed to be measured. An advantage of this method is that no interference with dissolved oxygen control is required. However, for relatively short periods between consecutive aerations of the reactor, the error of such a measurement is relatively high, and usually lowered values of the OUR are expected. This error depends on the PWM function parameters (T_{ON}, T_{tot}).
- The second regime on the other hand actively interferes with the dissolved oxygen control algorithm and causes inhibition of aeration for a selected period of time. Due to this non-aeration period, the dissolved oxygen value drops considerably, hence allowing a more accurate measurement of OUR.

The measurement of k_La requires the aerator to be switched off for a certain period of time in any case, since usually a greater decrease in dissolved oxygen is needed. Therefore, the k_La agent used for the determination of this parameter takes over control of the aeration process. When a suitable decrease in dissolved oxygen concentration is achieved, the aeration is switched on with a constant PWM parameters. Based on the previously measured OUR value, and on the rate of dissolved oxygen increase, the k_La parameter is evaluated and returned by the agent. The k_La value determined in this way corresponds to the currently set parameters of the PWM function.

Fig. 2 presents the schematic of the realized control system, taking into account the individual and in some cases independent software modules – that is – agents. All those module have been realized in a graphical programming environment LabVIEW and may either work as independent applications connecting with each other using TCP/IP protocol, or may be embedded in a single application. Those applications may either be implemented using an industrial PC, or may be embedded in a real-time operating system FieldPoint FP-2020 provided by National Instruments. This system is used as one of the possibilities to implement control algorithms on the pilot plant. Based on measurements obtained by those agents, another software agent is used to control the retention of biomass in the settler, hence influencing the mass balance of activated sludge biomass throughout the systems used in the presented experiments.

4 Experimental Case Studies

Actions of particular agents have been experimentally verified using the experimental pilot plant shown in Fig. 1. The biological reactor has been aerated with dissolved oxygen concentration set point equal to 1,5 [mg O_2/l]. The Control Agent, that is normally active, aerates the reactor when the measured oxygen concentration drops below the set point value. Actions of this agent are identical to the normal deterministic control system operation. However, this agent may be disengaged thus inhibiting the dissolved oxygen control loop. One possibility of such disengagement is a request from the k_La Agent to suspend aeration in order to perform experimental k_La determination. Fig. 3 presents an example of such actions. In the first phase, the Control Agent is disengaged periodically as can clearly be seen by the oxygen concentration dropping below the set point value. Such drops are necessary in order to accurately determine the k_La value. Data in Fig. 3 show three separate k_La experiments performed. Once the Control Agent's actions are no longer inhibited by the k_La Agent, dissolved oxygen concentration is maintained above the set point value as is required.

In most cases, the OUR Agent is not requiring any actions to be taken and passively monitors dissolved oxygen concentration. When a suitable period with no aeration is detected, the OUR value is estimated and returned by the agent. In some cases, this agent may take over the dissolved oxygen control loop and determine the OUR value more precisely, by disengaging the Control Agent, and thus allowing for a greater drop in oxygen concentration. On the other hand, OUR values that are passively determined by this agent are needed by the k_La Agent in order to estimate the correct value of k_La.

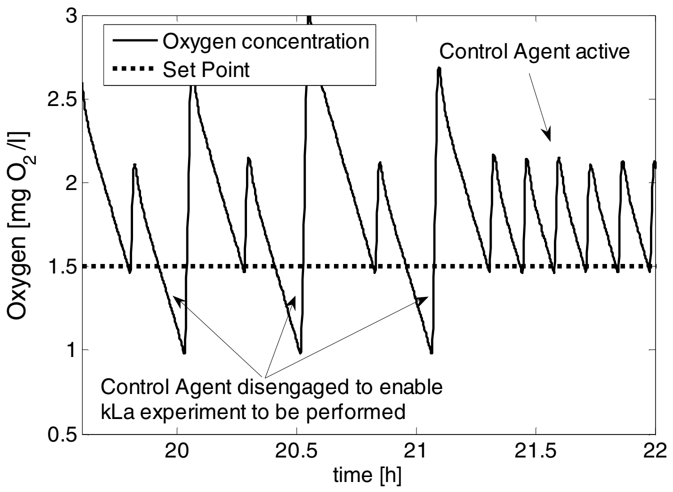


Fig. 3. Example of k_La Agent and Control Agent actions

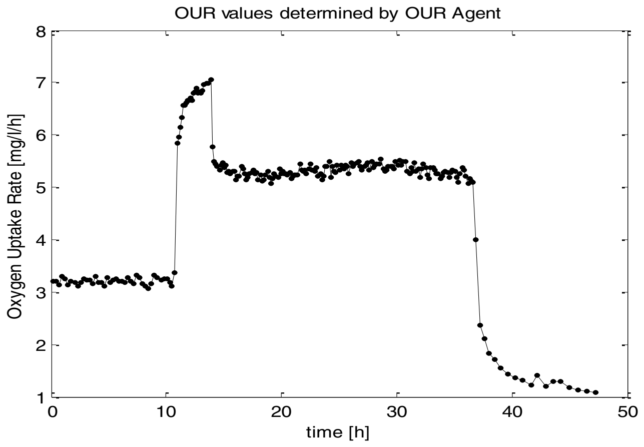


Fig. 4. OUR values measured by OUR Agent by means of passive monitoring of measurements

Fig. 4 presents an example of OUR values measured by the OUR Agent by passive monitoring of oxygen concentration. At first, the reactor is kept at a constant load of synthetic wastewater, resulting in a rather constant value of oxygen uptake rate around 3,2 [mg O_2/l/h]. At t = 10 [h], the load is drastically increased and than decreased to an intermediate value at around t = 15 [h] and kept constant afterwards. The visible slow variations in OUR value between t = 15 [h] and t = 37 [h] result from changes of biomass concentration due to growth and decay processes. At t = 37 [h] the dosage of synthetic wastewater has been stopped, resulting in a drastic drop in oxygen uptake by the biomass. The steady state value of OUR obtained with no load of wastewater represents endogenous respiration and results form the fact, that oxygen is needed to support living organisms even when no substrate for growth is available.

5 Concluding Remarks

The presented control system for a biological reactor is based on a Multi Agent System approach. Independent software modules (agents) realize their own goals and may use other agents to reach those goals. Additionally, agents are capable of inhibiting other agents in order to enable specific actions to be realized. Analyzing behavior of such control systems suggests that this approach greatly simplifies synthesis of control systems, creating systems that are easier to understand and implement with respect to classical control architectures. Moreover, the MAS-based approach introduced elements of the artificial intelligence such as cooperative distributed reasoning.

Future works will be connected with implementation of such a control system in the RSlogix programming environment for the Logix-series controllers.

Acknowledgments. This work was supported by the Polish Ministry of Science and Higher Education using funds for 2008-2010 under grant no. N N514 296335.

References

1. Wooldridge, M., Jennings, N.R.: Intelligent agents: theory and practice. The Knowledge Engineering Practice 10(2), 115–152 (1995)
2. Jennings, N.R., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. Autonomous Agents and Multi–Agent Systems, vol. 1, pp. 7–38. Kluwer Academic Publishers, Boston (1998)
3. Weiss, G. (ed.): Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (1999)
4. Knapik, M., Johnson, J.: Developing Intelligent Agent for Distributed Systems. Mc Graw–Hill, New York (1998)
5. Shamma, J.S. (ed.): Cooperative control of distributed Multi-Agent-Systems. John Wiley & Sons, The Atrium (2007)
6. Van Dyke Parunak, H.: A practitioners' review of industrial agent applications. Autonomous Agents and Multi-Agent Systems 3(4), 389–407 (2000)
7. Marik, V., McFarlane, D.: Industrial Adoption of Agent-Based Technologies. IEEE Intelligent Systems, 27–35 (January/February 2005)
8. Fletcher, M., Brennan, R.W., Norrie, D.H.: Modelling and reconfiguring intelligent holonic manufacturing system with Internet-based mobile agents. Journal of Intelligent Manufacturing 14, 7–23 (2003)
9. Kotak, D., Wu, S., Fleetwood, M., Tamoto, H.: Agent-based holonic design and operations environment for distributed manufacturing. Computers in Industry 52, 95–108 (2003)
10. Pechoucek, M., Marik, V.: Industrial deployment of multi-agent technologies: review and selected case studies. Auton Agent Multi-Agent Syst. 17, 397–431 (2008)
11. Choiński, D., Nocoń, W., Metzger, M.: Application of the Holonic Approach in Distributed Control Systems Designing. In: Marik, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 257–268. Springer, Heidelberg (2007)
12. Metzger, M., Polaków, G.: Holonic Multiagent-Based System for Distributed Control of Semi-industrial Pilot Plants. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS, vol. 4659, pp. 338–347. Springer, Heidelberg (2007)

13. Metzger, M.: Virtual controllers improve Internet-based experiments on semi-industrial pilot plants. In: Proceedings of the 16-th IFAC World Congress, Praga. Elsevier, Amsterdam (2005)
14. Nocon, W., Choinski, D., Metzger, M.: Web-based control and monitoring of the experimental pilot plant installations. In: Proceedings of the IFAC Workshop on Programmable Devices and Systems – PDS 2004, Krakow, pp. 94–99 (2004)
15. Polaków, G., Metzger, M.: ppPDC Communication Framework – A New Tool for Distributed Robotics. In: Carpin, S., Noda, I., Pagello, E., Reggiani, M., von Stryk, O. (eds.) SIMPAR 2008. LNCS (LNAI), vol. 5325, pp. 195–206. Springer, Heidelberg (2008)
16. Spanjers, H., Vanrolleghem, P.A., Olsson, G., Dold, P.L.: Respirometry in Control of the Activated Sludge Process. Principles. IWAQ, Scientific and Technical Report No. 7 (1998)
17. Spanjers, H., Patry, G.G., Keesman, K.J.: Respirometry based on-line parameter estimation at full-scale WWTP. Water Science and Technology 45(4-5), 335–343 (2002)
18. Baeza, J.A., Gabriel, D., Lafuente, J.: In-line fast OUR (oxygen uptake rate) measurements for monitoring and control of WWTP. Water Science and Technology 45(4-5), 19–28 (2002)
19. Third, K.A., Sepramaniam, S., Tonkovic, Z., Newland, M., Cord-Ruwisch, R.: Optimisation of storage driven denitrification by using on-line specific oxygen uptake rate monitoring during SND in a SBR. Water Science and Technology 50(10), 171–180 (2004)
20. Badino Jr., A.C., Cândida, M., Facciotti, R., Schmidell, W.: Improving k(L)a determination in fungal fermentation, taking into account electrode response time. Journal of Chemical Technology and Biotechnology 75(6), 469–474 (2000)
21. Soons, Z.I.T.A., Shi, J., Stigter, J.D., van der Pol, L.A., van Straten, G., van Boxtel, A.J.B.: Observer design and tuning for biomass growth and kLa using online and offline measurements. Journal of Process Control 18(7-8), 621–631 (2008)

Design and Implementation of LabVIEW-Based IEC61499 Compliant Device

Grzegorz Polaków and Mieczyslaw Metzger

Department of Automatic Control, Electronics and Computer Science
Silesian University of Technology, ul. Akademicka 16, 44-100 Gliwice, Poland
{grzegorz.polakow,mieczyslaw.metzger}@polsl.pl

Abstract. The method of IEC 61499 compliant device implementation with the National Instruments LabVIEW is proposed. The work focuses on these aspects of the tasks of the event generation and dispatching, which have no direct counterparts in the G language. A mapping of all the IEC 61499 concepts onto the G language concepts is described. Because of the limited multithreading support in LV (multithreading is possible but a number of parallel threads is fixed at the stage of the program development and compilation) it is needed to fit all the IEC 61499 defined functionalities in a fixed number of threads. A FIFO queue based dispatching algorithm is implemented, similar to the one used in the C++FBRT implementation. The ultimate objective of the work is the development of the FBLV run-time environment, which converts the LabVIEW compatible industry grade hardware into the IEC 61499 compliant device.

Keywords: distributed control, IEC 61499, holonic systems, LabVIEW, run-time environment.

1 Introduction

The first standard for holonic industrial systems is the IEC 61499 [1]. It took place of previously existing non-standardized products, however the market still lacks any successful industrial implementations. The most known run-time environment for the IEC 61499 is the FBRT developed by Holobloc [2], which is a part of the FBDK. The FBDK framework is designed primarily as a reference environment [3] as it exploits every aspect of the standard. The FBDK is developed using the Java programming language. There are not many industry-grade hardware implementing the Java Virtual Machine, which limits possible industrial use of the FBRT software. The process of industrial adaptation of the IEC 61499 advanced when the ICS Triplex [4] included the support for the standard in their ISaGRAF software suite. It allows to develop event-based applications, which may be targeted for the large set of supported operating systems. Drawbacks of this solution are: the lack of support for the purely industrial hardware, and controversial method of algorithm scheduling using the scan-based approach [5]. There are few more run-time environments implementing the norm compliant events and function blocks (see [6] for a brief survey), but usually those environments rely heavily on an underlying operating system. The only embedded environment known at the moment is the C++ conversion of the FBRT [7][8], which apparently did not get beyond the stage of research.

This paper presents the progress of the development of a run-time environment complying to the IEC 61499 norm (see [9] and [10] for reference). The environment is developed using the G language of the LabVIEW platform [11]. The G language is equipped with a wide set of capabilities, some of which are similar to the concepts of the norm (i.e. event queues, basic OOP). However, the G language constructs are general in nature, and using them to implement the norm compliant behavior requires much work. Proposed mapping of the concepts of the IEC 61499 norm onto G language constructs is the main contribution of this work. With the approach presented in this paper it is possible to program the LabVIEW compatible industry-grade hardware in the norm compliant way. The ultimate objective of the work described is the development of complete run-time environment (called from now on as a FBLV) which, when uploaded to a PAC (Programmable Automation Controller – see [11]), converts the PAC into the IEC 61499 compliant device.

It should be noted that the FBLV is not designed as a complete framework exploiting fully all the concepts of the standard. The goal of the implementation is to develop a run-time environment for industry-grade hardware, because the standard did not receive any attention from the biggest automation equipment manufacturers, which resulted in a lack of IEC 61499 compliant equipment (for now the most popular solution of this problem was uploading the FBRT to the Netmaster family controllers).

At the moment, preferred method of the FBLV device programming is by uploading an XML file generated by the FBDK, describing the function block network. However, in future it is expected that the FBLV device will be fully cooperative online as a part of system configuration. Such run-time environment would largely expand the set of the IEC 61499 compliant hardware by inclusion of all the PLCs and PACs made by National Instruments. An additional contribution of the work is the description of proposed single threading scheduling algorithm and low-level implementation of the norm defined FBs. This description can be easily adopted for other programming languages with limited multi threading capability.

2 General Concept

The G language used by the National Instruments LabVIEW development environment is a graphical language exploiting the concept of the function blocks similar to the FBD language of the IEC 61131 standard [12]. The G language was developed at first as a tool simplifying the data acquisition process, but nowadays it is more general. LabVIEW implements the concepts of loops, data structures, subprograms, GUIs, etc. – all of them realized in the form of dataflow driven block diagrams. The language is compilation-based, the diagrams are compiled into executable code, which may be targeted for various platforms, including FPGAs and PACs. The code is generated with a support for multithreading, parallel fragments of code are executed as separate threads. The most obvious drawback of the G is the complete lack of C-style memory pointers, which makes it impossible to construct, for example, linked list. The other problem is the limitation of the dataflow-based multithreading. While parallel loops are executed as concurrent threads, the number of threads is determined automatically (depending on the structure of the code and the capabilities of the target hardware) and fixed at the compilation stage. In effect, LV lacks the mechanism similar to the Java thread class – it is not possible to execute a given subprogram as a separate thread. However, LabVIEW is aware of the event-based programming concept. Typically, events are used in LabVIEW for GUI servicing, but it is possible to define user's own class of events. Such user-created classes of events are then dispatched with the LabVIEW built-in event manager in a G language construct called the Event Structure.

2.1 Mapping the Basic Concepts of IEC 61499 onto G Language

The standard defines three classes of devices, numbered from 0 to 2. The FBLV was designed to implement the functionality of the class 1 device. The class 0 was considered as too simple – it is more fitted for the intelligent sensors or actuators. For effective programming of the control algorithms, the support for FB instances is required, which leaves classes 1 and 2 under consideration. However, class 2 device, while more powerful, requires a compilation or an interpretation of a FB block algorithm given by an user. A run time-compilation of LabVIEW subprograms is not possible, a LabVIEW application has to be compiled and uploaded to the embedded device at once. An interpretation of the FB source code during run-time could be implemented, but it would lack the performance required in industrial use. In effect, the FBLV was decided to implement the functionality of class 1 device. The class 1 is well balanced between simplicity and capabilities, assuming that the library of FB types provided by the device is rich enough.

To keep the prototype implementation simple, it was decided that the FBLV will not provide a support for the multiple resources. The FBLV device accommodates a network of function blocks directly, which is equivalent to zero resources [9].

Events and scheduling algorithm. It is proposed to implement the events defined by the standard directly as the LabVIEW events serviced with the Event Structure. In consequence, the IEC 61499 events which are issued in the network of function blocks are stored in the LabVIEW internal event queue, which is working according to the FIFO principle. The IEC 61499 events are defined as the LabVIEW user events (distinct to the LabVIEW standard GUI events) carrying additional data – reference number of the block instance and the event input number at which the event arrived. Using the data carried by the event, the scheduling algorithm thread determines the proper reaction to the event, as shown in the Fig. 1.

The consecutive steps of the event dispatching process are as follows (numbered as in the Fig. 1):

1. The event is removed from the queue. Using the variables carried by the event, it is determined on which input of which of the existing FB instances the event is issued.
2. Using the table of the existing FB instances the FB type of the considered FB instance is read. Knowing the state of the FB instance (stored in the table of instances), the FB type, and the considered event input, the Event Dispatcher determines a chunk of code to be executed. The algorithm performs the task defined by IEC 61499 for the given input of a given FB type.

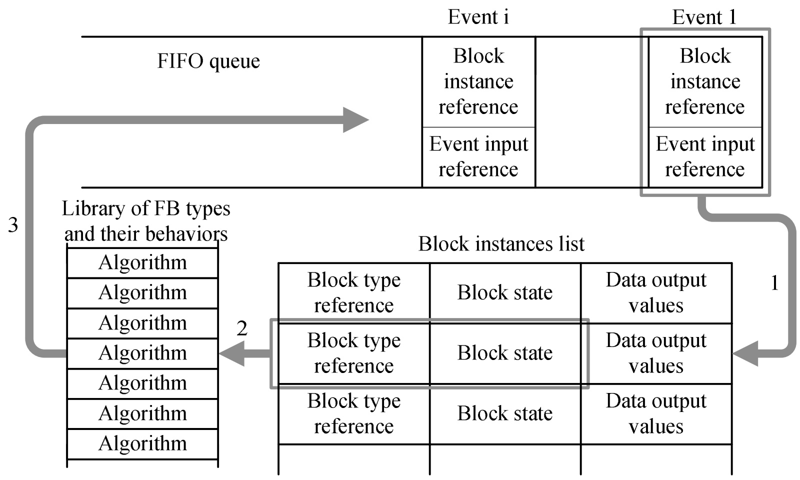


Fig. 1. The idea of the scheduling algorithm

3. If the task is supposed to result in issuing the new event, the proper entry is added to the queue of events waiting to be dispatched (after resolving the FB network and finding the destination FB instance and input).

A similar scheduling algorithm introduced in [6] by Zoitl et al. was named the Event Dispatcher. Due to this similarity, the scheduling algorithm proposed for the FBLV platform will be called the same. The work principle of the Event Dispatcher algorithm has a peculiar property: all the events issued in the FB network are serviced in the same processing thread, the events are dispatched sequentially. In effect, each of the events is eventually dispatched, events cannot be lost, as it is observable in the scan-based approach [13].

Block types. In the most of the existing implementations, the FB types are represented as the classes of a object oriented programming language (e.g. in FBDK). Algorithms executed by the events incoming to FBs are implemented as the methods of the classes. Although the LabVIEW implements the basics of the OOP, it is limited by the nature of the data-flow based programming, i.e. it lacks the dynamic thread creation. In effect, implementation of the FBs as the LabVIEW classes is unreasonable. Instead, a procedural programming based concept is introduced. Each of the FB type definitions is divided into basic procedures implementing the FB reaction to the events at the specific inputs. These procedures are spread and programmed as parts of the Event Dispatcher code. If a function block uses the ECC and its behavior depends on its internal state, a procedure additionally reads the state of the FB and executes a proper subroutine.

This distribution of the FB types behavior amongst the code of Event Dispatcher is the most important modification of the usual approach, resulting in the possibility of the FBLV development. The Event Dispatcher is simply the collection of basic algorithms, which are executed in an order defined by a FB connection network. It should be noted that the internal FB algorithms do not have to be developed exactly as the norm proposes – only the external behavior of the given algorithm has to be exactly as the standard defines. ECCs defined by the standard do not need to be implemented to the letter. ECCs are, in the case of the FBLV, less the programming tool, more a guideline for the programmers of the Event Dispatcher algorithms.

The algorithm collection stored in the Event Dispatcher is described by the global memory structure describing the implemented FB types, their names, inputs and outputs. This structure is the connection between the naming introduced by the standard and the pieces of code distributed in the Event Dispatcher.

Block instances. In the OOP-based implementations, FB instances of a given FB type are the objects of the given class. In the FBLV FB instances are stored as entries in the separate memory structure. The structure contains two fields describing properties of the instance: number of the FB type of which it is an instance, and the *variant* structure (similar to the non-typed memory pointer in C language) storing the state of the FB instance. The state consist of current values of data outputs of the instance, and any data specific to the instance (file handles, network socket handles, number of internal state according to the ECC, etc.). The creation of the FB instance consists of simple adding of the entry in the table and initiating its state.

Event and data connections. The connections between the FB instances are stored in an additional memory structure. A record of the structure contains four fields: the reference to the source FB and its connected output number, and the reference to the destination FB and its input number. There are two separate structures for the event connections and for the data connections. The structure of event connections is used directly by the Event Dispatcher, so it can find the correct destinations of newly induced events (step 3 in the Fig. 1). The structure of data connections is used by the algorithms, which find the source FB instance of the data values they require, so the correct entry containing the values can be found in the instances table.

Data types. The large part of the data types defined by the IEC 61131 standard (from where, they were adopted to the IEC 61499) have native equivalents in the G language. Integer and floating typed are currently implemented in the FBLV using the native types. STRING and WSTRING are easily implementable using the LabVIEW built-in string type. The IEC 61499 specific data types i.e. color, arrays and matrices have native equivalents. The time and date, user-defined and complex types are the most difficult, the proper implementation method is not determined at the moment. The TIME type bears the specific problem, as its resolution (i.e. 1μs) is out of LabVIEW capabilities – see Timed Events section below.

The most effective way of all the data types implementation is the development of the classes hierarchy, similar to the FBRT. The G language limited OOP support is sufficient enough for such an application.

2.2 Sources of Events

The presented idea of the Event Dispatcher assumes that all the events issued in the FB network are produced by the algorithms called during dispatching the previously existing events. Function blocks which are the sources of the events do not fall into this category, and they have to be implemented in other way. The two categories of such FBs are: the service interface blocks, and the blocks issuing the output events according to the pass of time. The solutions proposed below are basically a LabVIEW specific implementation of timed interrupts, which are the most fit for this task and used in the C++FBRT [14].

Timed events. The standard defines three FBs measuring the time, i.e. E_DELAY, E_CYCLE and E_TRAIN. Because the latter two may be implemented using the E_DELAY, it is actually the only time related FB requiring implementation. In the FBLV it is proposed to implement the second event dispatcher specifically for this task. The Timed Events Dispatcher is not based on a FIFO queue as the original one, instead its buffer holds the unordered list of the events which are scheduled to be issued by E_DELAY blocks in a future. A separate thread (i.e. parallel *while* loop) cyclically checks the buffer with a minimal achievable in LV period, removes the events due in the last period, and adds them to the main Event Dispatcher. The LabVIEW specific problem is the lack of widely compatible precise hard real-time timers. The only real-time loop provided by the G language, which is compatible with all the National Instruments hardware, has the resolution of 1ms. More precise timers are available on selected platforms only. At the current stage of FBLV development, for the compatibility, it is assumed that the 1ms resolution is enough, but it surely may be not precise enough in industrial applications. As a consequence part of the TIME data type capacity is ignored.

Service interface blocks. A similar idea has to be implemented for service interface blocks. Blocks which send information and do not expect reception confirmation may be treated as any other blocks, and implemented with the main Event Dispatcher. However, reception of incoming data which results in issuing new events has to be serviced in a separate software thread. Therefore, opening of the network sockets is done within the main Event Dispatcher, handles to the sockets are stored as the state variable in the table of FB instances. A separate thread cyclically checks opened listening sockets. If any data arrived and is stored in the buffers of network adapters, the thread reads the data, assigns it to the outputs of a proper FB instance, creates the IND event, and adds it to the main event queue.

3 Implementation

The following sections present few fragments of already developed source code, which may serve as the reference for the developers of similar solutions. The code presented is developed with the G language, but it is described thoroughly, so the general idea is adoptable in the other programming languages.

3.1 Event Dispatcher and Function Block Algorithms

The source diagram of the Event Dispatcher thread is shown in the Fig. 2. It is relatively simple, because it uses the LabVIEW built-in Event Structure. At first, the class of user events is created (1), which carries two variables: the FB block instance number and the event input number. The event class is registered (2), so the events of the class may be dispatched in the Event Structure (3) placed in the infinite While Loop (4). The diagram also illustrates the concept of FB type specific algorithms. Using the FB Instance number (5) the subVI (6) finds the FB type number of the instance in the

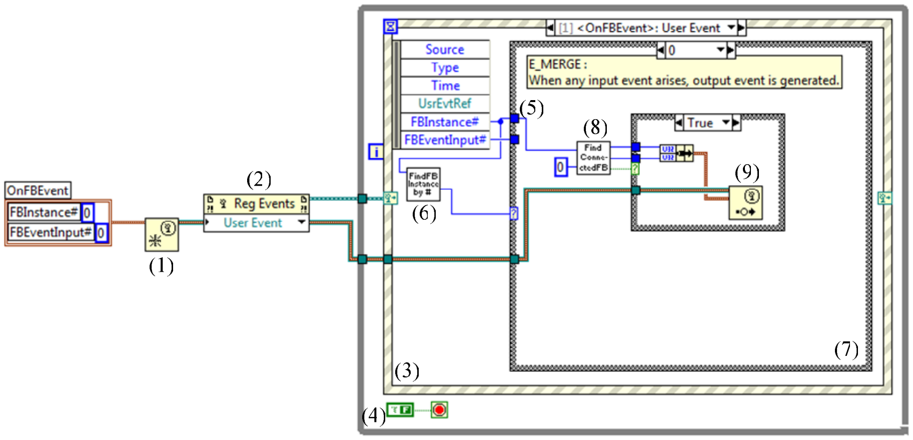


Fig. 2. The source diagram of the scheduling algorithm and the E_MERGE algorithm

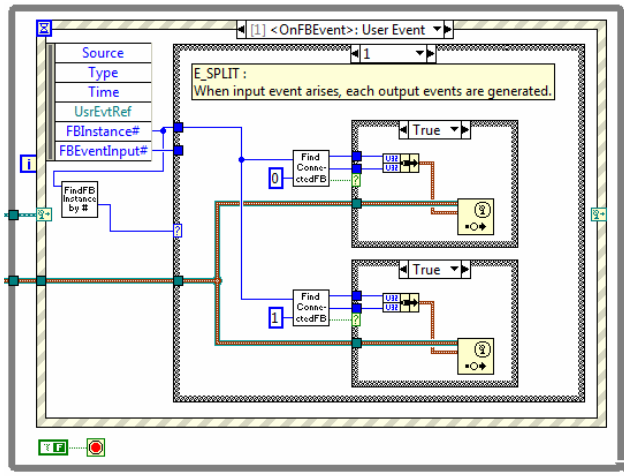


Fig. 3. The E_SPLIT algorithm

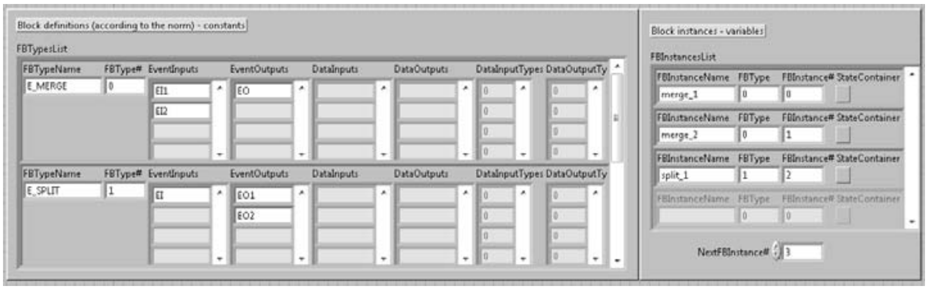
instances table. The FB type number selects one of the cases from the Case Structure (7). In the Fig. 1 the implementation of E_MERGE FB type is shown. If the FB type number equals zero (in the FBLV zero is assigned to the E_MERGE type) the diagram fragment shown in figure is executed. The algorithms issues the output event regardless of the input event number which induced the algorithm. At first, the search for the FB instance connected to the output of the serviced FB instance is performed (8). If found, the event at its proper event input is added to the Event Dispatcher queue (9).

For comparison, in the Fig. 3 the internal algorithm of the E_SPLIT FB type is shown. When an event arises at the input of the E_SPLIT instance, a search for the instances connected to both the event outputs of the E_SPLIT instance are performed. Events at inputs of those FB instances are then added to the event queue. All the blocks defined by the standard are implemented in a similar way in the Event Dispatcher code.

The algorithms embedded in the Event Dispatcher are supplemented by the global array of clusters (a cluster is a LV counterpart of a C structure) describing all the implemented FB types and their inputs and outputs. The cluster is shown in the graphical form in the Fig. 4. The same figure shows also the instances table. In the figure three FB instances are declared, two of the E_MERGE type and one of the E_SPLIT type.

Additionally, there also exist arrays of clusters not included in the figures, i.e. the table of event and data connections (containing the pairs of FB instances numbers and event/data input/output numbers), and the table of defined data types.

At the current stage of the implementation progress, the only method of interaction with the run-time environment is the GUI pictured in the Fig. 5. FB instances and connections are presented in the textual form. FBs and connections can be created and deleted. To help in the debugging tasks, the history of the events fired is provided. In


Block definitions (according to the norm) - constants
Block instances - variables

FBTypesList

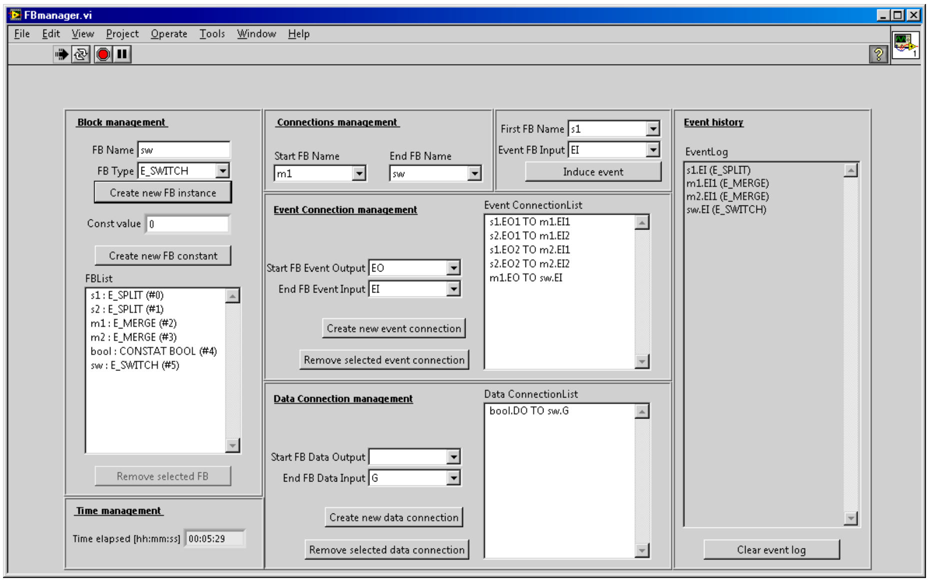
FBTypeName	FBType#	EventInputs	EventOutputs	DataInputs	DataOutputs	DataInputTypes	DataOutputTy
E_MERGE	0	EI1	EO			0	0
		EI2				0	0
						0	0
						0	0
E_SPLIT	1	EI	EO1			0	0
			EO2			0	0
						0	0
						0	0

FBInstancesList

FBInstanceName	FBType	FBInstance#	StateContainer
merge_1	0	0	
merge_2	0	1	
split_1	1	2	
	0	0	

NextFBInstance# 3

Fig. 4. The description of implemented FB types and the list of currently existing FB instances


FBmanager.vi
File Edit View Project Operate Tools Window Help
Block management
Create new FB instance
Create new FB constant
Remove selected FB
Time management
Connections management
Induce event
Event Connection management
Create new event connection
Remove selected event connection
Data Connection management
Create new data connection
Remove selected data connection
Event history
Clear event log

FB Name sw
FB Type E_SWITCH
Const value 0

FBList
s1 : E_SPLIT (#0)
s2 : E_SPLIT (#1)
m1 : E_MERGE (#2)
m2 : E_MERGE (#3)
bool : CONSTAT BOOL (#4)
sw : E_SWITCH (#5)

Time elapsed [hh:mm:ss] 00:05:29

Start FB Name m1
End FB Name sw

First FB Name s1
Event FB Input EI

Start FB Event Output EO
End FB Event Input EI

Event ConnectionList
s1.EO1 TO m1.EI1
s2.EO1 TO m1.EI2
s1.EO2 TO m2.EI1
s2.EO2 TO m2.EI2
m1.EO TO sw.EI

Start FB Data Output
End FB Data Input G

Data ConnectionList
bool.DO TO sw.G

EventLog
s1.EI (E_SPLIT)
m1.EI1 (E_MERGE)
m2.EI1 (E_MERGE)
sw.EI (E_SWITCH)

Fig. 5. The user interface of the current pilot implementation of the FBLV

further development a support for XML files is planned, so the files generated by the FBDK could be parsed into FBLV specific FB network description. It should be noted that the stage of the FB network creation and the stage of Event Dispatcher work are not separated. Both the threads work parallel, enabling the support for the dynamic reconfiguration of the system. It is possible due to the complete lack of structure compilation – all the connections between the blocks are stored and resolved with the use of dynamic memory structures.

4 Concluding Remarks and Future Work

In this work the method of IEC 61499 compliant device implementation with the National Instruments LabVIEW is described. The work focuses on those of the tasks of the event generation and dispatching, which have no direct counterparts in the G language. Because of the limited multithreading support in LV (multithreading is possible but a number of threads is fixed at the stage of the program development) it is needed to fit all the IEC 61499 defined functionalities in a fixed number of threads. Currently the FBLV run-time environment is at the development stage, but since all the main concepts are formulated the full functionality is achievable. The current prototype implementation uses desktop PCs as a hardware platform, and the interaction with the outer world is HMI based. In future, it is planned to change the target platform to the PAC hardware, which will effectively enable the creation of industrial grade IEC 61499 compatible run-time environment. The most challenging task is the implementation of timed events with the 1μs resolution, which will require the use of hardware clocks existing in the PAC platforms.

It should be noted that the performance and reliability of the FBLV approach are still not verified experimentally. The comparison of the FBLV with the other IEC61499 approaches is planned as soon, as the FBLV development advances enough.

The approach based on the dynamic memory structures and FIFO queue enables the FBLV to change the structure of the FB network without stopping the main Event Dispatcher thread. In effect, the FBLV is capable of a dynamic reconfiguration, implementation of the RECFB FB type proposed by [15] is considered in the future.

Acknowledgments. This work was supported by the Polish Ministry of Science and Higher Education using funds for 2008-2010 under grant no. N N514 296335.

References

1. IEC, Geneva. IEC 61499-1: Function Blocks – Part 1 Architecture (2005)
2. HOLOBLOC Inc.: HOLOBLOC Inc. Webpage, `http://www.holobloc.com`
3. Hall, K.H., Staron, R.J., Zoitl, A.: Challenges to Industry Adoption of the IEC 61499 Standard on Event-based Function Blocks. In: 5th IEEE International Conference on Industrial Informatics, vol. 2, pp. 823–828. IEEE Press, New York (2007)
4. ICS Triplex: ISaGRAF Webpage, `http://www.ics.triplex.com`

5. Vyatkin, V., Chouinard, J.: On Comparisons of the ISaGRAF implementation of IEC 61499 with FBDK and other implementations. In: 6th IEEE Conference on Industrial Informatics, pp. 289–294. IEEE Press, New York (2008)
6. Zoitl, A., Strasser, T., Hall, K., Staron, R., Sünder, C., Favre-Bulle, B.: The past, present, and future of IEC 61499. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 1–14. Springer, Heidelberg (2007)
7. Zoitl, A.: Development of an IEC 61499 based embedded control platform and integration in a distributed automation system. Master's thesis, Vienna University of Technology (October 2002)
8. Rumpl, W.E., Auinger, F., Dutzler, C., Zoitl, A.: Platforms for Scalable Flexible Automation Considering the Concepts of IEC 61499. In: Mařík, V., Camarinha-Matos, L.M., Afsarmanesh, H. (eds.) IFIP Conference Proceedings, vol. 229, pp. 237–246. Kluwer B.V., Deventer (2002)
9. Vyatkin, V.: IEC 61499 Function blocks for embedded and distributed control systems design. ISA, Research Triangle Park (2007)
10. Lewis, R.: Modelling control systems using IEC 61499. Applying function blocks to distributed systems. IEEE, London (2001)
11. National Instruments Website, `http://www.ni.com`
12. IEC, Geneva. IEC 61131 Programmable controllers – Part 3: Programming languages (1993)
13. Cengic, G., Ljungkrantz, O., Akesson, K.: Formal Modeling of Function Block Applications Running in IEC 61499 Execution Runtime. In: Proceedings of the 11th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2006, Praque (2006)
14. Sünder, C., Rofner, H., Vyatkin, V., Favre-Bulle, B.: Formal description of an IEC 61499 runtime environment with real-time constraints. In: 5th IEEE International Conference on Industrial Informatics, vol. 2, pp. 853–859. IEEE Press, New York (2007)
15. Rooker, M.N., Sünder, C., Strasser, T., Zoitl, A., Hummer, O., Ebenhofer, G.: Zero Downtime Reconfiguration of Distributed Automation systems: The εCEDAC Approach. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 326–337. Springer, Heidelberg (2007)

Holonic-Based Environment for Solving Transportation Problems

Michał Gołacki, Jarosław Koźlak, and Małgorzata Żabińska

Department of Computer Science,
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
{kozlak,zabinska}@agh.edu.pl, michal@golacki.com.pl

Abstract. In this paper, a system based on the concept of holons for solving transportation problems is presented. Such a system focuses on the possibility of comparing the quality of solutions offered by the system with the results obtained from classical algorithms for commonly used sets of test problems. These problems were modified in order to take into consideration the fundamental features offered by the holonic approach.

1 Introduction

The application of methods which efficiently realise transport requests has a crucial significance for the competitiveness of companies. For the sake of the high complexity of algorithms for solving transport problems, different heuristic approaches are used. Widely examined classic transport problems such as VRP (Vehicle Routing Problem) and its different variants (e.g. *Vehicle Routing Problem with Time Windows* - *VRPTW*, *Pickup and Delivery - PDPTW, Pickup and Delivery with Time Windows* - *PDPTW*) operate on a simplified model of the world, which does not take into consideration essential problems that transport companies experience in reality. It would be especially necessary to take into account heterogeneity of transport units (trucks, trailers, and drivers). It seems to be useful to apply approaches based on holons because such a complex unit may be treated as a single element at most of its actions, having at the same time the capability of reorganising, removing and replacing its elements.

The goal of a proposed system is to solve transport problems starting from the static problem, hitherto tasks of the dynamic problem extended with the use of holonic agents. The classic problem consists in finding a solution for a set of transport requests, when the whole pool of requests is known at the start of computations. The solution makes a set of routes for transport units. All units are the same, and a transport network is a complete graph. A dynamic version of the problem means that a pool of requests is not known at the beginning of computations. Subsequent requests appear during the course of the simulation. It requires from a system a continuous adjustment to changes. It is not necessary that a graph of a transport network is complete, but it may be, to preserve a more likelihood to the classic problem. The next step is to take into consideration different elements and to treat transport units as complex structures, built with elements such as trucks, trailers, drivers, etc. Representatives of component classes may differ among themselves regarding properties and attributes: e.g. power and velocity of trucks, load of trailers, types of loads to carry, drivers' qualifications, their knowledge of routs, languages, experience or a number of hours worked without any break.

2 State-of-the-Art

Holonic multi-agent systems. Combining the approach based on holons with the multi-agent systems has been a subject of a significant amount of research as the concept is promising however at the same time raises many important questions. They concern the structure and rules of how the holon organisation functions, in particularly, the relationship of the agent on the one hand and its limitations caused by being a part of larger structure - holons, the motivation of an agent to join holons and the protocols of actions needed for leaving and joining a holon.

To enable and facilitate the organisation of agents in holons, it is useful to describe the know-how of agents in a way such as to enable them to identify the agents possessing the capacity and features needed for them. In [13] a concept of capacity and its application in holonic multi-agent systems is presented. In [7] an adaptive architecture for multi-agent holonic systems which uses immunological algorithms is presented. This approach was applied to the management of a group of mobile robots playing football in an FIRA league.

The holonic agent approach may be applied to many domains and particularly to modelling the functioning of companies and supply-chains, in Distributed Control Systems [4] and management of a shipping company [3].

Transportation problem solving. Different approaches are used for solving transportation problems: evolutionary algorithms [8], tabu search ([10]), Squeaky Wheel Optimisation [11] or multi-agent approach. The multi-agent solutions are based on Contract Net Protocol [14] and for optimisation simulated trading [6] or DCSP (Distributed Constraint Satisfaction Problem) [5,12] are used.

The first multi-agent transportation system based on holonic approach was Teletruck [3]. As opposed to models used for solving classical transportation problems, it took a higher flexibility and diversity of elements into consideration – different kinds of trucks, trailers, containers, drivers as well as their features and characteristics. The main ideas of an algorithm for allocating requests was similar to the solution used in MARS system [6].

3 Concept of Multi-agent System

The purpose of the work was to built a system which is functionally close to TeleTruck and to concentrate on the analysis of its possibilities in comparison with existing classic algorithms. Next, we want to carry out a comparison of results gained by such a system with classic solutions of transportation problems. For analysis we chose PDPTW and extensions introduced by us, which are related to holonic approach. The system enables a solution of static and dynamic problems; it may operate in an environment based on Euclidean distance and a graph representing road connections between locations.

3.1 Agents' Description

In the constructed system the following main types of agents exist: Dispatcher Agent (DA), Driver Agent (DRA), Truck Agent (TKA), Trailer Agent (TRA), and Transportation Unit Agent (TUA).

Dispatcher Agent. DA plays a role of a manager in the system. It controls the work and life cycle of Transportation Unit Agents (TUA), as well as distributes transport requests between them in the best possible way. In the system only one Dispatcher Agent exists.

$$DA = (G_D, S_D, K_D, A_D) \tag{1}$$

where:

G_D - goal which is to minimise the function:

$$G_D = \Sigma_{i=1}^{n} Cv^i + \Sigma_{i=1}^{n} Cd^i TD^i \tag{2}$$

where n – number of $TUAs$, Cv^i – cost of exploitation (mainly cost of amortisation and driver's salary) of TUA^i, TD^i – distance travelled by TUA^i, Cd^i – average cost of TUA i for travelling a distance unit .

S_D – state — $S_D = \{LocAgent_j\}, j = 1 \ldots n$ estimated location of all agents j,

K_D – knowledge, $K_D = \{Reqs, StatReqs, Env\}$, where $Reqs$ – information about requests, $StatReqs$ – information about status of requests, which assigns, to each of them, a value from the set {received, rejected, allocated, picked–up, delivered}, Env - information about transport network,

A_D - actions to be realised, $A_D = \{AllocReq, GetAgPos, SendAgPos\}$, $AllocReq$ – auction of a request and its allocation to a Transportation Unit, $GetAgPos$ – get info about agent positions, $SendAgPos$ – send info about agent positions.

The more detailed descriptions of actions, taking into consideration elements of the agent having a direct influence on them (left parts of expressions, before arrows) and being modified as a result of the action (right parts of expressions, after arrows) are as follows:

$AllocReq$: $Reqs \times StatReqs \rightarrow StatReqs$
$GetAgPos$: $LocAgent_j \rightarrow LocAgent_j$
$SendAgPos$: $LocAgent_j \rightarrow another\_agent$ (communication action, sending information to the other agent)

Truck Agent. Truck Agent i (TKA^i) represents a main element (vehicle) of TUA which is able to travel on its own together with the driver but without a load capacity:

$$TKA^i = (S^i_{TKA}, K^i_{TKA}, A^i_{TKA}) \tag{3}$$

where:

S^i_{TKA} – state: $S^i_{TKA} = \{TKAf^i, Loc^i, Bel^i_{TUA}, Res^i_{TUA}$, where $TKAf^i$ – TKA features (among others power and acceptable types of connections with a trailer), Loc^i –location, Bel^i_{TUA} – belonging to the TUA, Res^i_{TUA} – reservation for TUA and the due time.
K^i_{TKA} – knowledge $K^i_{TKA} = \{Info^i_{DRA}, Info^i_{TRA}\}$: information about neighbouring (in the same depot) drivers ($Info^i_{DRA}$) and trailers ($Info^i_{TRA}$),
A^i_{TKA} – actions $A^i_{TKA} = \{Join^i_{TUA}, Disj^i_{TUA}\}$: $Join^i_{TUA}$ – join to the TUA, $Disj^i_{TUA}$ – disjoin from the TUA.

Trailer Agent. Trailer Agent i (TRA^i) represents everything which is used by the truck to carry the load:

$$TRA^i = (S^i_{TRA}, K^i_{TRA}, A^i_{TRA}) \tag{4}$$

where:

S^i_{TRA} –state, $S^i_{TRA} = \{TRAf^i, Loc^i, Bel^i_{TUA}, Res^i_{TUA}\}$, where $TRAf^i$ – TRA features (among others kind of load, maximal and current load, acceptable types of connections with a truck), Loc^i –location, Bel^i_{TUA} – belonging to the TUA, Res^i_{TUA} – reservation for TUA and the due time.
K^i_{TRA} – knowledge, $K^i_{TRA} = \{Info^i_{DRA}, Info^i_{TKA}\}$, information neighbouring (in the same depot) drivers ($Info^i_{DRA}$) and trucks ($Info^i_{TKA}$).
A^i_{TRA} actions: $A^i_{TRA} = \{Join^i_{TUA}, Disj^i_{TUA}\}$, where $Join^i_{TUA}$ – join to the TUA, $Disj^i_{TUA}$ – disjoin from the TUA.

Driver Agent. Driver Agent i (DRA^i) represents a driver of a truck:

$$DRA^i = (S^i_{DRA}, K^i_{DRA}, A^i_{DRA}) \tag{5}$$

where:

- S^i_{DRA} –state, $S^i_{DRA} = \{DRAf^i, Loc^i, Bel^i_{TUA}, Res^i_{TUA}\}$ where $DRAf^i$ – DRA features (among others hitherto performed working hours, licences for driving given kinds of trucks and shipment, route knowledge, foreign languages knowledge, preferences concerning intermediate points), Loc^i – location, Bel^i_{TUA} – belonging to the TUA, Res^i_{TUA} – reservation for TUA and the due time.
- K^i_{DRA} – knowledge, $K^i_{DRA} = \{Info^i_{TKA}, Info^i_{TRA}\}$: information about neighbouring (in the same depot) trucks ($Info^i_{TKA}$) and trailers ($Info^i_{TRA}$),
- A^i_{DRA} actions: $A^i_{DRA} = \{Join^i_{TUA}, Disj^i_{TUA}\}$, where $Join^i_{TUA}$ – join to the TUA, $Disj^i_{TUA}$ – disjoin from the TUA.

Transportation Unit Agent. Transportation Unit Agent i (TUA^i) manages the transport unit which realises transport requests. Together with other units it participates in auctions of transport requests, and then it realises the assigned requests.

$$TUA^i = (G^i_{TUA}, P^i_{TUA}, Loc^i_{TUA}, S^i_{TUA}, K^i_{TUA}, A^i_{TUA}) \tag{6}$$

where:

G^i_{TUA} agent's goal to obtain the maximal possible income:

$$G^i_{TUA} = Cv^i + Cd^i TD^i \tag{7}$$

where n – number of TUs, Cv^i – cost of exploitation (mainly cost of amortisation and driver salary) of TUA^i, TD^i – distance travelled by $TUA\ i$, Cd^i – average cost of TUA^i for travelling a distance unit .

P^i_{TUA} - planned travel route, contains list of locations to be visited, performed pick up and delivery operations and holon reorganisation,

Loc^i_{TUA} current position of TUA,

S^i_{TUA} - agent's state, $S^i_{TUA} = \{AlReqs^i, HolStruc^i, Res^i_{DRA}, Res^i_{TRA}, Res^i_{TKA}\}$, where $AlReqs^i$ – allocated requests, $HolStruc^i$ – information about holon structure and component agents, reserved drivers (Res^i_{DRA}), trucks (Res^i_{TRA}) and trailers (Res^i_{TKA}),

K^i_{TUA} - agent's knowledge $K^i_{TUA} = \{Reqs^i, Env^i\}$ comprising $Reqs$ - set of requests, Env - information about environment (transport network),

A^i_{TUA} - actions of the TUA:

$$A_{TUA} = \{ReqPart^i, Move^i, PickUp^i, Delv^i, HolCrt^i, HolDis^i, HolRorg^i\}, \tag{8}$$

where $ReqPart$ - participation in auctions of transport requests, $Move$ – transfer between locations, $PickUp$ - loading, $Delv$ - unloading, $HolCrt$ – holon creation, $HolDis$ – holon dissolution, $HolRorg$ – holon reorganisation.

The more detailed description of actions is as follows:

$ReqPart$ - ($AlReqs^i$, Env^i) $\rightarrow$ ($AlReqs^i$, P^i_{TUA}),
$Move$ – $Loc^i_{TUA} \rightarrow Loc^i_{TUA}$
$PickUp$ - ($AlReqs^i$, Loc^i_{TUA}) $\rightarrow AlReqs^i$
$Delv$ - ($AlReqs^i$, Loc^i_{TUA}) $\rightarrow AlReqs^i$
$HolCrt$ – ($HolStruc^i$, Res^i_{DRA}, Res^i_{TRA}, $Res^i_{TKA}\}$) $\rightarrow$ ($HolStruc^i$, Res^i_{DRA}, Res^i_{TRA}, $Res^i_{TKA}\}$)
$HolDis$ – ($HolStruc^i$, Res^i_{DRA}, Res^i_{TRA}, $Res^i_{TKA}\}$) $\rightarrow$ ($HolStruc^i$, Res^i_{DRA}, Res^i_{TRA}, $Res^i_{TKA}\}$)
$HolRorg$ – ($HolStruc^i$, Res^i_{DRA}, Res^i_{TRA}, $Res^i_{TKA}\}$) $\rightarrow$ ($HolStruc^i$, Res^i_{DRA}, Res^i_{TRA}, $Res^i_{TKA}\}$)

3.2 Optimisation Algorithm

Allocation of requests between agents is performed on the basis of the Contract Net Protocol.

The transportation unit agents (completed or not) announce cost estimations of request realisations. These estimations may be based on capabilities of transportation units (parameters of the unit, its location and already accepted request) or the estimated possibilities of holon creation by them, which perform auctions of offers of given components which could make a holon (section 3.3 and section 4).

Requests are assigned to these agents which may realise them with the smallest increase of costs. Next, optimisation of solution is carried out – every given interval of time each TUA tries get rid of offers which are less profitable for it from the point of view of costs and it sends them to auctions. In this way the requests may be transferred to other TUAs (section 4).

The holon can also reorganise itself (section 3.3) to obtain a better configuration of components to fulfil requests allocated to it at lower costs.

3.3 Holon Organisation

Cooperation between agents consists in completion of transport units needed to execute requests and their common realization, as well as minimization of costs of a given set of requests. The applied model assumes that agents resign from a part of their independence only. The whole holon is represented by Tranport Unit Agent - TUA, which communicates with the whole society. It has the right to assign resources, plan, and negotiate with other agents on the basis of plans and aims of component agents. At the same time a task of TUA is to replace elements of the holon or to join and separate them.

Holons are created when a new request is received, if it enables a cheaper realisation. Moreover, one can assume that some holons already exist at the beginning of simulation. The created TUA does not have any elements of holon at the beginning, so it has to be created from the basis. A task is sent to all types of elements, and then a unit waits for their replies. After it receives them, the best offers are chosen, and a cost of task execution is computed and then an offer is sent to Dispatcher. In cases where a positive answer is received, elements are informed and a holon is created. Other elements receive a negative reply. Such an answer is sent to all elements of a holon, when the holon is not chosen for the task. Then, such a unit stops to exist. If a holon is created, its elements loose a part of their autonomy. They will not be able to receive information about new requests from units other than the unit which is the manager of their holon and they will not be able to send answers to such requests.

Reorganisation of a holon may be done in cases of new requests which cannot be realized by a given unit.

Dissolving of a holon takes place when a transportation unit performs all entrusted tasks. Resources are freed and they may join other holons, whereas TUA waits for subsequent requests. If they appear, it tries to collect the next transportation unit and starts to realise them.

4 Realisation

The system is divided into some modules. They are, viz. computational module, layer controlling a simulator including user's interface, means of visualisation and tests generator. In particular, the computational module comprises agents of the system which are responsible for finding a solution. Tests generator is able to change static tests into dynamic ones, to increase the number of available transport bases as well as modify a transport network. Agents use JADE [2] platform (Java Agent Development Framework, Version 3.3) and communicate among themselves undertaking actions to solve a given problem.

Realisation of agents' tasks in the system is related to communication between them. Main tasks of agents are following:

– assignment of a new request, when holon (TUA) with requests already exists,
– assignment of a new request, when holon (TUA) already exists, but without any requests,
– assignment of a new request, when holon (TUA) does not exist and it is necessary to create it (creation of TUA),
– lack of requests for holon (TUA) after finishing all tasks, dissolution of the holon.

The assignment of a new request, when holon exists, but requests have not yet been assigned to it, is presented below. This situation is connected with the reorganisation of TUA (fig. 1). TUA receives a request and verifies its capability to realise the request. If such a possibility exists, Dispatcher Agent accepts TUA. Otherwise, if TUA is not appropriate to realise the request, it tries to reorganise itself. Let us assume, that an element which is not suitable to realise the given request is a trailer. Then trailers, which have not joined any TUA yet, are informed and send their offers. After choosing the best offer, the offer for

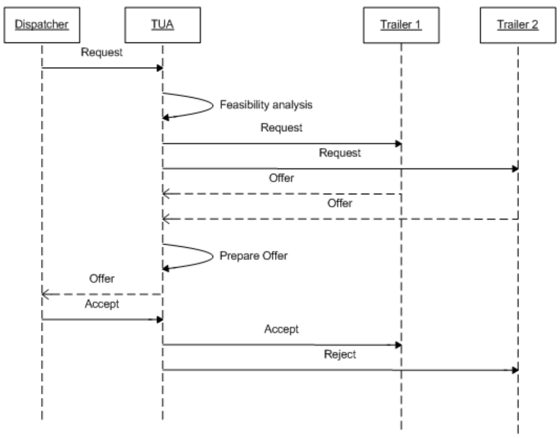


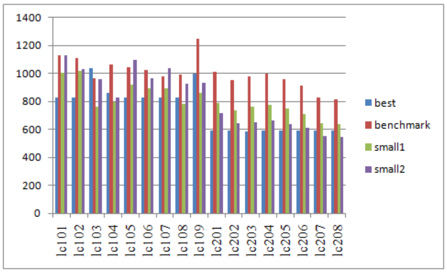
Fig. 1. Sequence of actions: a new request and TUA reorganisation

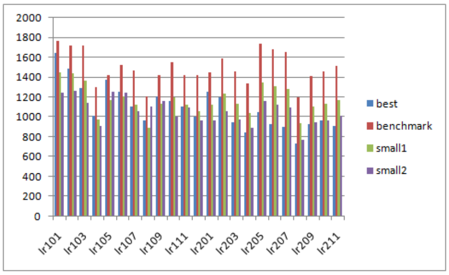
realising the given request is sent to Dispatcher. The chosen TUA transfers the information to the trailer which replaces the one present until now in the holon.

5 Experiments and Results

To verify the quality of solutions obtained, sets [1] of tests prepared by Li and Lim [10] have been used. Tests have been modified to include holons. Standard tests comprise of PDPTW with a different number of points of pickup and delivery. Results obtained for problems with 100 pickup and delivery points are presented below. Problems may be categorised into classes where points of requests are located in clusters (LC), distributed uniformly (LR) or comprise points grouped both in clusters as well as distributed uniformly (LRC). Moreover lxx are problems with narrow time windows, whereas 2xx - with wide ones. Since we examine the activity of a transport company which possesses a pool of vehicles, the key parameter concerning criteria of optimisation is distance, which limits fuel consumption of available vehicles, and makes easier to be included in subsequent requests. Therefore the number of used vehicles does not play a primary role.

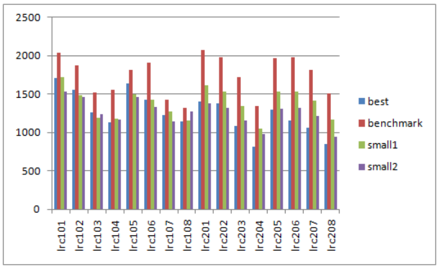
In subsequent charts (fig. 2a, fig. 2b and fig. 3a) results (costs) obtained for the above-mentioned groups of problems are shown. In each figure there are results for narrow and wide time windows. A modification introduced to the tests was the possibility of using trailers of different load capacities. Subsequent bars, viz. *best*, *benchmark*, *small1*, *small2* present costs of realisation of requests for the best known solutions of problems, our solutions for vehicles of load in agreement with those used in a benchmark, whereas for *small1*, and *small2* we have at our disposal trailers of different capacities - those of *small1* are smaller then those of *small2*. In particular, if for a given transport problem, the capacity of vehicles in a test set was x, then for *small1* problems we have at our disposal 10 vehicles for each capacity: x, 3x/4 and x/2, whereas for *small2*, 10 vehicles for each capacity: x/2, 3x/8 and x/4. Costs are related to travelled distance and capacity of the used trailer; they are calculated as: $CD = (CD_{DR} + CD_{TK} + CD_{TR})/3$, where CD_{DR}, CD_{TK}, CD_{TR} – average travel costs for a distance unit of Driver, Truck



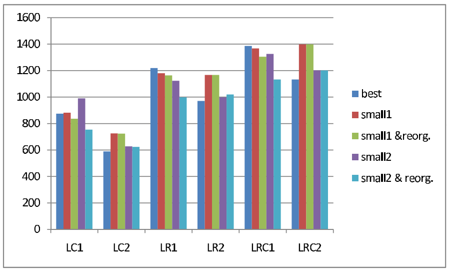


a) b)

Fig. 2. Results: Results for LC (a) and LR (b)problems with narrow and wide time windows



a)



b)

Fig. 3. Results: (a) LRC problems with narrow and wide time windows (b) Families of results for 100 request points, without and with holon reorganisation groups

and Trailer and each of these travel element costs CD_x are calculated as follows: where: $CD_x = (1/3 + (2 * Cp_{max})/(3 * Cp_d))$, Cp_{max} – maximal capacity of the given element, Cp_d – capacity of vehicles used in a standard solution.

The presented results show that costs of travel for benchmarks versions (that is, capacities in consent with those used to obtain best known solutions of benchmarks) usually only slightly differ from the best known solutions. However, using smaller vehicles in *small1* and *small2* problems, it was possible to decrease costs substantially, which is a proof that the whole capacities of vehicles were not usually used in solving test problems. It also shows advantages of flexible planning, when we have trailers of different capacities at our disposal, because it may limit costs significantly.

Aggregated results for main groups of benchmarks are presented in fig. 3b. There are LC1 and LC2, – clusters with narrow and wide time windows, LR1 and LR2 – uniformly distributed with narrow and wide time windows, LRC1 and LRC2 – clusters and uniformly distributed with narrow and wide time windows. For each case the results for different sets of vehicle capacities as well as for the algorithm without and with holon reorganisation are taken into consideration.

We can notice that the holon reorganisation significantly decreased costs of solutions for cases with narrow time windows. For the cases with wide time windows the improvement did not occur, it is related to the fact that capacities of vehicles exceeded considerably the needs.

6 Conclusions

In this paper, the results obtained using our holonic system for modelling and optimising transportation requests were presented. We described the pilot version of the application. Subsequent work will focus on the introduction of new algorithms for constructing routes and increasing the flexibility of algorithms for constructing holons. The next domain of work will take into consideration different kinds of critical situations and in particular, traffic jam formations, route closing as well as breakdown of vehicle or other holon elements. These aspects have een analysed in our various works in this area [9], and ind the future be integrated with the holonic approach.

References

1. Benchmarks - Vehicle Routing and Travelling Salesperson Problems, http://www.sintef.no/static/am/opti/projects/top/
2. Java Agent DEvelopment Framework, http://jade.tilab.com/
3. Burckert, H.-J., Fischer, K., Vierke, G.: Transportation scheduling with holonic MAS - the TELETRUCK approach. In: Third International Conference on Practical Applications of Intelligent Agents and Multiagents (PAAM 1998) (1998)
4. Choinski, D., Nocon, W., Metzger, M.: Application of the holonic approach in distributed control systems designing. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 257–268. Springer, Heidelberg (2007)
5. Dorer, K., Calisti, M.: An Adaptive Solution to Dynamic Transport Optimization. In: Proceedings of the AAMAS 2005 industry track, Utrecht, The Netherlands (2005)
6. Fischer, K., Muller, J., Pischel, M.: Cooperative Transportation Scheduling: an Application Domain for DAI. Applied Artificial Intelligence, pp. 1–33 (1996)
7. Hilaire, V., Koukam, A., Rodriguez, S.: An adaptative agent architecture for holonic multi-agent systems. ACM Trans. Auton. Adapt. Syst. 3(1), 1–24 (2008)
8. Homberger, J., Gehring, H.: Two evolutionary meta-heuristics for the vehicle routing problem with time windows. INFORMS Journal in Computing 37(3), 297–318 (1999)
9. Konieczny, M., Koźlak, J., Żabińska, M.: Multi-agent crisis management in transport domain. In: Proceedings of ICCS 2009. LNCS. Springer, Heidelberg (accepted for publication, 2009)
10. Li, H., Lim, A.: A Metaheuristic for the Pickup and Delivery Problem with Time Windows. In: Proceedings of 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2001), Dallas, USA (2001)
11. Lim, H., Lim, A., Rodrigues, B.: Solving the Pick up and Delivery Problem using ”Squeaky Wheel” Optimization with Local Search. In: Proceedings of American Conference on Information Systems, AMCIS 2002, USA (2002)
12. Neagu, N., Dorer, K., Calisti, M.: Solving Distributed Delivery Problems with Agent-Based Technologies and Constraint Satisfaction Techniques. In: Dist. Plan and Schedule Management, AAAI Spring Symp., USA. The AAAI Press, Menlo Park (2006)
13. Rodriguez, S., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: An analysis and design concept for self-organization in holonic multi-agent systems. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS, vol. 4335, pp. 15–27. Springer, Heidelberg (2007)
14. Smith, R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computer, 1104–1113 (December 1980)

Holonic Manufacturing Paint Shop

Morten Lind[1], Olivier Roulet-Dubonnet[1], Per Åge Nyen[2], Lars Tore Gellein[2], Terje Lien[1], and Amund Skavhaug[3]

[1] Department of Production and Quality Engineering, The Norwegian University of Science and Technology, Norway
morten.lind@ntnu.no
[2] SINTEF Raufoss Manufacturing, Norway
[3] Department of Engineering Cybernetics, The Norwegian University of Science and Technology, Norway

Abstract. In pursuit of flexibility and agility within discrete manufacturing, the surrounding logistics and handling processes of a paint shop is under construction as a laboratory prototype application. Holonic Manufacturing seems to be a promising strategic paradigm and architecture to use for a system characterised by production logistics and control. This paper describes the physical devices to be used; the desired functionality; and the basic logic control designed. Additionally, the ideas for holonification based on the already designed logic control is presented.

1 Introduction

Distributed decisions and coordination of autonomous sections in manufacturing have been around as long as complex manufacturing. By complex manufacturing in our context, we mean dealing with multiple complex products and extensive sharing of manufacturing equipment across different simultaneous product variants, i.e. with frequent changeovers and reconfigurations on the shop floor. In the recent decades there has been focus on the flexible automation of these entities. The enabling technologies for this may be traced back to the birth of numerical control and computational intelligence in the 1950s.

Holonic Manufacturing is a paradigm for pervasive manufacturing automation, ranging from (and integrating with) the lowest level of real time shop floor control and all the way up to company or even corporate level. It covers most aspects of manufacturing, be it machine to machine cooperation or order to production department interaction.

The concept of a *Holonic Manufacturing System* (HMS) date back to the early 1990s when the *Intelligent Manufacturing Systems* (IMS) initiative set out a project with that name. The term *Holon* was coined by Arthur Koestler[1], some 40 years ago, for capturing the dualistic capabilities of autonomy and cooperativeness withing a single entity. The concept was found suitable to encompass the entities, physical as well as abstract, in manufacturing control and management.

There exist some architectures for Holonic Manufacturing Systems, such as PROSA[2] and ADACOR[3]. PROSA is strictly a reference architecture and introduces the central concepts of basic holons: order, product, resource, and staff holons. High level scenarios illustrate the interactions of the different holon types. ADACOR is also an architecture, but with a different naming of the holon types. Notably the ADACOR supervisor holon differs from the PROSA staff holon, in that it formally coordinates the dynamics of holon aggregation and subordination. Leitão and Restivo applies the ADACOR architecture to a (partially simulated) machining and assembly workshop in several papers, see e.g. Leitão and Restivo[4,5].

Two standards are highly relevant for Honlonic Manufacturing on very different levels: IEC61499 and FIPA; cf. Mařík et al.[6]. The IEC61499 standard regards a function block structuring of design and code for low level control oriented holons and their interactions. At the higher level, FIPA is a standard for ontology based agent communication.

The laboratory prototype described in this paper is a part of IntelliFeed, a cooperative project between the research institutions of the authors and industrial partners, supported by the Norwegian Research Council. Relevant and related projects also based at our research institutions are RAMP and CREAM; both part of the CRI NORMAN research program[7].

The particular laboratory prototype system we present here has its origin in the recognition that much manual labour is associated with the materials handling around paint shops in manufacturing industry. In itself, the full or partial automation of such systems will have a good potential for reducing trivial manual labour. But the potential is extended further if the automation is flexible and responsive, enabling the paint shop system to integrate with other parts of the entire manufacturing system.

This paper is organised as follows. In Sect. 2 the physical devices and their layout is described. Sect. 3, presents aspects of a holonification of elements and control. Discussion and future challenges is presented in Sect. 4.

2 Application Overview

In this section we describe the laboratory layout and the associated physical devices.

A real paint shop roughly consists of a painting system, a part upload station, a part download station, a painting carrier upload station, a painting carrier download station, and an overhead conveyor system. The painting carriers are hanging from the conveyor trolleys and are transported through all the stations and the painting system.

Our current goal is automation of the processes and materials handling at the part upload station. In the future, the other stations will undergo equivalent automation projects in a successive manner.

2.1 Laboratory Application Overview

A sketch of our initial laboratory system setup, currently used for simulation with QUEST, is shown in Fig. 1. The transport AGVs and a *Parts Arrangement*

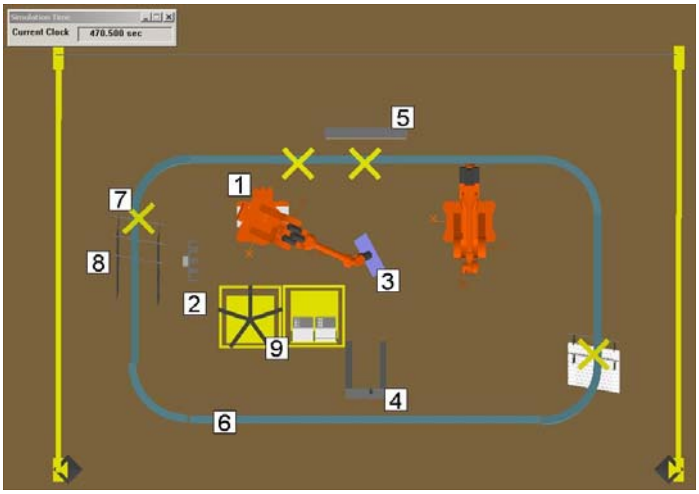


Fig. 1. Overview layout of a planned laboratory system. 1) *Upload Robot*, 2) *Upload Tool Rack*, 3) *Mounted Upload Tool*, 4) *Upload 3D Vision System*, 5) *Backlight Screen*, 6) *PnF Overhead Conveyor Track*, 7) *PnF Switch*, 8) *Painting Carriers*, and 9) *Local Part Storage Carriers*.

Station are not seen. The setup is planned for a Power-and-Free (PnF) overhead conveyor, though the conveyor type has not been decided for yet. As will become clear, a PnF conveyor is definitely to wish for when automating the uploading of parts to the painting carriers.

The robots we have available in the upload area are two Nachi SC15F, which are floor mounted within reach of each others and the conveyor track. Both are equipped with a 6D force sensor at the wrist. One of the Nachi robots have a modified controller, giving a direct $100 - 200\,Hz$ interaction with the servo controller using UDP over Ethernet. The other Nachi controller can easily be modified likewise.

In addition we have an ABB IRB 2400 with an S4C+ controller, also located in the laboratory but well outside the conveyor area. The ABB robot is connected to the rest of the system over an RS232 serial communication link. It has its own standard robot controller programmed in the complex RAPID control language. The program execution is hard real time, but interaction can only be through parameter modifications before real time execution.

2.2 Part and Painting Carrier Handling

The part uploading is performed by a robot which has access to parts from a local part storage. The upload robot picks a part from one of the local storages and then attaches the part onto the presented painting carrier on the conveyor. Given the part and painting carrier type, as well as known painting carrier capacity state, the support system will be able to control the robot to attach the part at some free site on the painting carrier.

The painting carriers may have very different geometries and have a load capacity from one to several tens of parts. Frequently each painting carrier type may match different part types; this has, of course, a high impact on the sequencing of part batches. Some painting carrier types are adjustable and can host a whole family of parts types. The automatic handling of painting carriers is not a part of the initial laboratory system. Manual change of painting carriers will be performed, when need be.

The process of localising the attachment points on some of the simpler painting carriers, with the Scorpion 3D vision system from Tordivel, has already been implemented and verified. For stable identification and analysis of all painting carrier types, it may be necessary with more vision system or other sensory systems.

To ensure mechanical stability of the painting carrier under identification, analysis, and upload, a controlled clamping mechanism will be implemented underneath the PnF switch at the upload station.

2.3 Conveyor System

The conveyor system has not been decided yet. This decision is of cardinal importance for a lot of other hardware decisions, as well as the control logic.

We concentrate our efforts around the PnF conveyor, simply because of the severe implications which would arise by using the more common *Chained Trolley* conveyor[8]. The main difficulty with a chained trolley conveyor is that all trolleys travel at constant speed at all times. Thus either the parts must be uploaded onto the painting carriers in motion, or the painting carriers must be sidetracked to a small PnF conveyor loop with a buffer for reattachment to the main conveyor.

2.4 Painting Process Scheduling

For a given paint process setup, with parameters for rate of paint added and geometric configuration of parts, it is desirable to use the maximum speed of the conveyor while still meeting the paint quality requirements. Or, for given speed of the conveyor and geometric configuration of parts on the carriers, it is desirable to minimise the rate of paint added, while meeting the required paint quality, thus minimising the amount of wasted paint.

Upload and download capacity and equipment utilisation makes up a delicate trade-off. This depends on the pertinent part type, painting carrier type and painting process parameters. In a simple batch controlled system, where only one part type and one painting carrier type is on-line at any time, there will often be under-utilisation in one or more of the upload station, the download station, and the painting system.

If mixing of parts within each painting carrier, or in the sequence of painting carriers, is allowed for, more freedom is given to the optimisation and the problems of under-utilisation can be remedied somewhat. The implication is a much more complicated logistics around uploading and downloading.

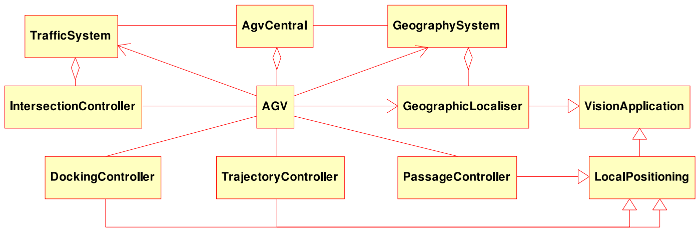


Fig. 2. A class diagram showing some important aspects and subsystems related to the AGV system

2.5 AGV System

Figure 2 shows a class diagram illustrating some important aspects of a multi-AGV system for material transport. Multiple AGVs are being built at our laboratory and the software system is in the design phase. The physical AGVs and AGV control system is based on earlier experience with a prototype AGV and vision based positioning system.

The conceptual AGVs, being hosted entirely on the physical AGVs, are central components in the AGV system. We initially assume that all of the AGV localisation functionality will be based on vision applications. This is based on earlier successes with developing a vision based AGV localisation system.

The traffic system gathers information from and supplies information to the client AGVs about viable paths in the whole of the roaming area. It will be the the coordinator for pre-reservation of long term trajectories of the AGVs, whereas the AGVs themselves can react to short term (emergency) trajectory interferences. Specialised intersection controllers at known congested areas are regulatory rather than guiding in their relation with the approaching AGVs.

The geography system is the general global localisation system for the AGVs. It is demanded that the cameras of the geographic localiser servers cover the entire area where the AGVs may roam, and can serve the client AGVs with real time location information.

Specialised conceptual controllers for local positioning and control may be defined to serve purposes relevant to situations or tasks, partly external to the AGV system. Examples, relevant to our laboratory system are docking, trajectory, and passage controllers.

3 Holonification

Based on the initial plans, layouts, and experiments with uploading of parts, a logic control of a paint shop has been designed. The given design is more or less a traditional hierarchical control design.

In this section we sketch our ideas for holonification of the logic control system and the physical devices and layout in the previous sections. Ideally the designed explicit logic control will emerge out of the holonic control system currently under development.

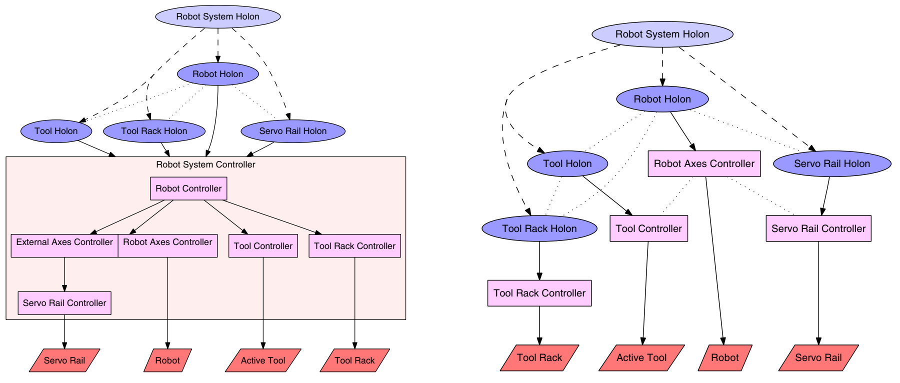
We are not suggesting that we have a complete design for the holonic system, but this is a description of our initial thoughts and ideas on how a holonic architecture will be applied in the control and management system.

3.1 Resource Holons

The resources, holons associated with physical devices, are related to the set of devices discussed in Sect. 2. This need not be a direct mapping, and indeed the devices described in Sect. 2 give too little detail to identify the entire set of resource holons.

The Robots. plays a central role to the part handling processes. They are the devices with the hardest real time aspects which we try to holonify. Normally the real time aspects of the motion control is integrated with other real time tasks inside the commercial controller, and closed to the intervention in a broader system, cf. Fig. 3(a). This should not be so in a deeply holonified system, since it enables only a virtual high granularity of the holonified control; this is illustrated in Fig. 3(a), by the low level holons all addressing the integrated robot controller for accessing their logical control functionalities. The device controllers' functionalities are hierarchically organised inside the commercial controller.

What is usually meant when referring to *a robot* in a manufacturing work station is the physical arm, the robot controller, and a whole range of various peripheral devices and their controllers, i.e. the whole *robot system*. These peripherals may consist of tools available for the robot, external (rail or gantry) axes for the robot, and even the whole range of sensors external to the robot arm. This is often well justified due to the *black hole* nature of commercial robot controller, in which coordination with external devices is possible only if these devices are integrated into, and controlled by the robot controller. The real time



(a) Centralised robot control system in vendor controller.

(b) Distributed holonic robot control system.

Fig. 3. Traditional integrated and holonic robot systems in automation. Holons are shown in blue colours and controllers and physical devices are shown in red colours.

aspects of the entities integrated into the commercial controllers is a good reason for the tight integration, but is an example of both types of *lock-in* processes described by Mařík et al. [6].

We have made an effort in separating the motion control of the mechanical arm from the control of the peripheral devices. This will enable us to perceive the real robot, i.e. without tools, external axes, etc., as a holonic device, giving a higher true holonic granularity in the holonic system. This is illustrated in Fig. 3(b) where the holons access their physical controllers directly, and where the device controllers are allowed peer-to-peer interaction for hard real time performance.

The robots locations, motion capabilities, and work space envelopes are of high importance to the order and product holons. Even for a homogeneous set of robots with the same product capabilities, a slight difference in location relative to the process location, may give a major impact in process performance. This will be even more expressed in a system of heterogeneous robots. These relations may not be a disadvantage, but rather an advantage, especially if the batch layouts and the scheduler functionalities allow for mixing on-line part and painting carrier types.

We are planning to have two different robot systems in the initial laboratory system. One is the system around the central upload robot and another system around the less described arrangement robot. The arrangement station is where bin picking is taking place, picking single parts out of gross storage carriers and placing them structured to order on local storage carriers, possibly directly on top of AGVs.

The Parts Arrangement Station is the location where parts are taken from gross storage containers and transformed into a pickable arrangement in a local storage carrier. The whole station, once robotised with bin picking capabilities, will be in itself a complex holonic system, if the supplier of the bin picking software and hardware allows it. It will definitely be so, in case we endeavour to implement it in-house.

For the initial phases, however, we must envision the absence of robotisation of this station, thus leaving the bin picking a manual operation. This is by no means in contradiction with the holonic manufacturing thought. On the contrary, one of the forces of the HMS paradigm is that an operator need not be treated or represented differently from any other resource or staff holon. The physical interaction with an operator is slightly different, though the capabilities should not be inferior to that of a robotised processing entity.

The direct interaction of this station with the paint shop system is to the orders and schedulers. Product holons for parts play a role in identifying methods for parts, and possible handling process information. Product holons for the transport carriers play another role of specifying the arrangement of parts into the carrier and possibly some process information for installation of the parts.

The Robot Tools are mountable on the robot arm and customised to handle specific parts. The entire set of tools available to one or more robots, or other handling device, must cover the whole range of parts to be handled by the robots.

In case of multiple tools, they must be organised in a tool rack, to be on-line changeable by the robot system. There will be a tool holon for each tool, but typically only the ones mounted on a robot at any given time will be active.

Fixtures, be it active or passive, may be considered as tools as well, and need not be at a fixed or active location all the time. In that sense, they are no different from what we normally refer to as tools; tools just just differ by being mountable on a robot end effector.

The tool set available to a robot system is of high importance to the product and order holons. A given order holon needs to be able to request from the robot system that it use a certain tool or tool capability, namely one that is compatible with one of the process alternatives in the pertinent product holon.

The Vision Systems planned for the initial laboratory system are comprised by the following four quite different types:

Painting Carrier Vision System: This is the 3D vision system already implemented with Scorpion 3D from Tordivel. It analyses for the points and directions of interest for a empty painting carrier presented to its cameras. This information is used by the upload robot system.

Part Localiser Vision System: Localises the parts for picking positions and orientations in the transport carriers (possibly the AGVs) in the local part storage area at the upload robot system. It is not to be considered a bin picking system, since the parts are at least semi-structured in the transport carriers.

Passage Vision System: The vision system used to guide the AGVs to cross intersection(s) of the transport route(s) with the overhead conveyor system. It is quite simply a matter of determining, in advance, when there will be passage where, and for how long.

Bin Picking Vision System: Used for identifying parts lying unstructured in gross storage carriers or containers.

This list of vision systems disregards the various dedicated and specialised AGV location vision systems, which are considered private to the AGV system.

The integration of the vision systems into holonic vision applications is a matter of letting orders interact with them in various ways, such as reserving for usage, configuring for detailed applications, and querying for or subscribing to information. But most vision system usage is expected to take place on a lower holonic level, integrated in a more numerical and detailed way with the direct consumers of the information, such as robot systems and AGVs.

The Conveyor System. As mentioned earlier, we are not yet sure what type of overhead conveyor system we will install, hence we know little about the controller capabilities and properties. The overall conveyor controller interaction will, however, be controlled by a holon.

In case of a PnF conveyor it will be relevant to find out if the PnF stations can be directly controlled, and thus be integrated with a physical device holon. Otherwise a logical device holon must be set up to access the main conveyor controller. If tracking of the trolleys is possible, it may be relevant to model each trolley as a resource holon. However passive, such trolley holons will at least be responsive to queries regarding arrival time, as well as associated painting carrier and order.

The AGV System. The holonic AGV system is under development, but is developed as a generic AGV system rather than a custom made system for the laboratory paint shop.

It may well be the case that the AGV system exposes its *products*, i.e. the transport and part carrier capabilities and capacities. This is a necessary information to access for the orders and schedulers in the paint shop system. There may be various alternative procedures for creating transport and configuration orders in the AGV system from the paint shop system. One is to send transportation requests to one or more AGV centrals, receiving an offer after negotiations internal to the AGV system. Another, lower level alternative is to directly create a transport order in the paint shop system and then have the pertinent paint shop order holon interact with relevant individual AGVs and AGV system schedulers, to settle a suitable contract. Both alternatives should be offered, since they may have their strengths and weaknesses in different situations.

3.2 Order Holons

The orders in the system contain the static data relevant for the associated order holons. We think of the orders as a hierarchy where the highest level is a production plan, being a quite simple list of parts for production during the day. Typically, in manufacturing systems today, the production plan for the day has been determined over night, or some days in advance, by the company ERP system. In a batch oriented system such as a paint shop, the sequencing and scheduling of the production plan is done heuristically by the operators at the paint shop. All parts produced in the painting systems we know of, are produced for intermediate storage, so the parts produced are used no earlier than the following day.

An illustration of a possible division of orders for our laboratory case is shown in Fig. 4.

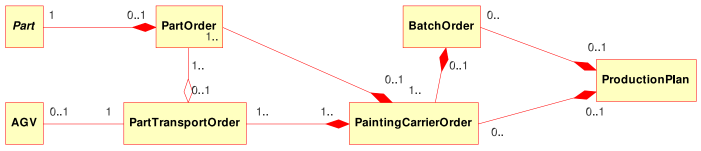


Fig. 4. Overview of aggregations and compositions among the order holon classes

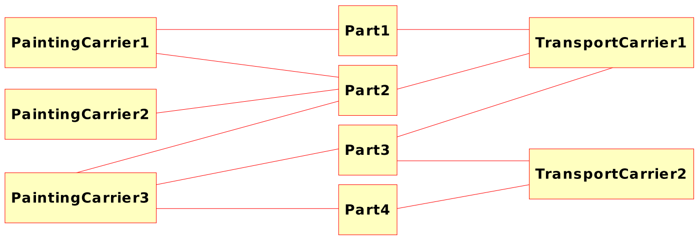


Fig. 5. Illustration of a match between painting and transport carrier types and part types in a simple example with four part types and three carrier types

3.3 Product Holons

The product holons for the upload station must contain a specification of how to attach a compatible set of parts onto a given painting carrier type. If similarity of parts and optimality of the painting process allow for it, certain painting carrier types will be able to host a range of part types. This matching is exemplified in Fig. 5.

The identical problem of matching of carriers and parts arises for transportation purposes. Shown also in Fig. 5 is an example of a transport carrier type to part type match.

In the cases described, the product holons are very static and persistent structures. This will be the case with predefined mixing of part types on the same carrier type. But to be truly flexible, the mixing should be possible to decide for in an on-line manner, i.e. some scheduling functionality may have an on-line or real time interaction with the uploader system. In this case some volatile product holons will be configured ad hoc for a single painting or transport carrier, or configured for the sake of some number of carriers; e.g. for mixing and completing a set of batches.

3.4 Staff Holons

Of staff holons we most certainly will have schedulers. In fact, one of the first overall control mechanisms we should experiment with, may be to emulate the decision system of the responsible operator in the real paint shops. The responsible paint shop operator has a fundamental input the painting day plan from the company ERP system. He makes on-line scheduling decisions based on a variety of parameters and feelings, which we should make an effort in investigating.

In addition to the vaguely interacting staff holons from PROSA[2] we embrace the more directly interacting supervisor holons from ADACOR[5]. We believe both aspects are good to have in separate entity types, whereas Leitão and Restivo[5] seems to suggest that the supervisor holon is more than a staff holon, and replaces it. The staff holons hold the responsibility for planning, long term scheduling, and global optimisation. And the supervisor holon, under normal system operation conditions, should have responsibility for and control of more local aspects of short term scheduling, optimisation, holon structures, and orchestration.

The distinguishing difference between staff and supervisor holons, in our interpretation, is that staff holons are available as a service and help to other holons, whereas decisions by supervisor holons have to be respected by other holons. Thus staff holons can be thought of as finding solutions to problems and supervisor holons to be executors and enforcers of such solutions. As per design of the supervisor holons by Leitão and Restivo[5], they lose their powers when disturbances arise, regaining them again when it is possible to restore local order. These views are not in contradiction with the PROSA and the ADACOR architectures.

4 Discussion and Future Work

While we will proceed with implementing the holonic control and management system in the near future, further development and implementation of hardware and device specific control software will take place in parallel.

One might say that we are still only in the analysis phase of developing the holonic manufacturing system for our laboratory prototype. We feel well into that phase, and have begun considering high level design issues. One pressing issue that we currently have much emphasis on is choosing middleware and platform.

Inspired by such works as Shin et al.[9], one viable path to follow is to start without a platform and use CORBA as middleware. This will enable us to progress fast by implementing our own platform using the Python Programming Language. Python is a language and platform with which we have good experience regarding both the management and soft real time domains of software. The direct implementation of the holons based on CORBA references invocations does not really support a fully flexible holonic platform. However, the platform we develop can be revised into a FIPA compliant one, adding complex communication abilities where and when it is needed.

Another path to follow might be more traditional, like the applications of ADACOR[5], where Leitão and Restivo use the FIPA compliant, Java based JADE agent platform. In order to apply our experience with Python we would use the FIPA compliant, Python based SPADE[10].

5 Concluding Remarks

At the current state of design of the laboratory paint shop prototype, we have gained such confidence in the HMS paradigm, that it will be used for the prototype, and recommended to the industrial partners in the project. The HMS paradigm has turned out to be a major initiator in bringing our minds from the offices towards the laboratory.

References

1. Koestler, A.: The Ghost in the Machine. Hutchinson, London (1967)
2. Brussel, H.V., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems: PROSA. Computers in Industry 37(3), 255–274 (1998)
3. Leitão, P.: An Agile and Adaptive Holonic Architecture for Manufacturing Control. PhD thesis, Department of Electrotechnical Engineering, Polytechnic Institute of Bragança (January 2004)
4. Leitão, P., Restivo, F.: ADACOR: A holonic architecture for agile and adaptive manufacturing control. Computers in Industry 57, 121–130 (2005)
5. Leitão, P., Restivo, F.: Implementation of a Holonic Control System in a Flexible Manufacturing System. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38(5), 699–709 (2008)
6. Mařík, V., Fletcher, M., Pěchouček, M.: Holons & agents: Recent developments and mutual impacts. In: Mařík, V., Štěpánková, O., Krautwurmová, H., Luck, M. (eds.) ACAI 2001, EASSS 2001, AEMAS 2001, and HoloMAS 2001. LNCS, vol. 2322, p. 233. Springer, Heidelberg (2002)
7. Lien, Welo, Nyen, Schütz, et al.: WP1: Technology and Trend Monitoring, White Papers. SINTEF Technology and Society (August 2007)
8. Kay, M.G.: Material Handling Equipment Taxonomy (1999)
9. Shin, J., Park, S., Ju, C., Cho, H.: CORBA-based integration framework for distributed shop floor control. Computers & Industrial Engineering 45(3), 457–474 (2003)
10. Escriva, M., Palanca, J., Aranda, G., García-Fornes, A., Julian, V., Botti, V.: A Jabber-based Multi-Agent System Platform. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 1282–1284. ACM, New York (2006)

Development of a Holonic Free-Roaming AGV System for Part Manufacturing

Olivier Roulet-Dubonnet[1], Morten Lind[1], Lars Tore Gellein[3], Per Åge Nyen[3], Terje Lien[1], and Amund Skavhaug[2]

[1] Department of Production and Quality Engineering, The Norwegian University of Science and Technology
[2] Department of Engineering Cybernetics, The Norwegian University of Science and Technology
[3] Department of Production Engineering, SINTEF Technology and Society

Abstract. This paper presents an Automated Guided Vehicle (AGV) system under development and its industrial background as a support system for an automated paint department. The focus is on demonstrating how the holonic architecture can be used to implement a flexible AGV system. The system is composed of autonomous AGV holons who cooperate, directly or as groups, with other holons such as robot holons, vision-system holons and order holons to produce the real parts. The holonic architecture is described in detail and example use-cases presented.

1 Introduction

A niche for small and medium enterprises is to produce individually tailored products or services. If the production is done on a large scale, this is called mass customization[1,2] and requires highly flexible automated production systems. The new generation of free-roaming Automated Guided Vehicles (AGV) can play an important role in this pursuit for flexible automation. AGVs are ground robots, usually used for transportation purposes on the manufacturing shop-floor. Taking fully advantage of the free-roaming AGVs require an integration into the manufacturing systems and a flexibility that the hierarchic, sequential systems currently implemented on most shop-floors do not offer.

Holonic manufacturing[3] is a promising architecture for the development of the new kind of flexible manufacturing systems needed. Holonic manufacturing is a highly distributed control paradigm that promises to handle frequent changes and disturbances successfully[4]. It combines features of both heterarchic and hierarchic organizational structures and is based on the concept of autonomous cooperating agents, called 'holons'. Holonic manufacturing is related to multi-agent systems (MAS), a paradigm derived from the distributed artificial intelligence field, but has some key differences[5]:

– Holonic manufacturing is a concept while MAS is a concept and a technology. It is possible to implement holonic manufacturing using MAS technology.

– A holon can represent simultaneously a whole and a part of the whole: a holon can be composed of several lower level holons.
– Some holons per definition represent and contain physical devices while agents are normally software components.

AGV systems are, by nature, distributed and, as such, we see the holonic paradigm as a good candidate for the architecture of AGV systems. By AGV system we mean one or more physical AGVs and the control and communication software that enable them to receive orders, schedule and execute them.

A systematic review of published papers on material handling in manufacturing systems was performed, with a focus on holonic systems. There are many publications on holonic manufacturing systems but only a few concentrate on material handling. In one of the first papers on the subject, Liu[6] proposes a distributed holonic architecture with one staff holon, where all service requests are sent to and handled by the AGVs themselves. Srivastava[7] presents an approach for conflict-free shortest path, minimum time motion planning and deadlock avoidance for AGV systems. It also presents an architecture for AGV systems which is influenced by its focus on zone algorithms. The architecture is partially reused in our research project. Babiceanu, in his PhD thesis[8], proposes a holonic-based control system for automated material handling systems. His thesis focuses on scheduling and he shows that the results, obtained by running the holonic algorithms, are close to the optimal solution. Most publications on transport systems focus on AGVs running on fast tracks with fixed crossing nodes and empty pathways[6,7,9]. They do not investigate the case of free-roaming AGVs driving in a stochastic environment together with other vehicles, manufacturing artefacts and humans.

This paper presents an Automated Guided Vehicle (AGV) system under development and its industrial background. The focus is on demonstrating how the holonic architecture can be used to implement a flexible AGV system. The system is composed of autonomous AGV holons who cooperate, directly or as groups, with other holons such as robot holons, visions-system holons and order holons to produce the real parts. The holonic architecture is described and example use-cases presented.

The remainder of this paper is organized as follow: Section 2.1 presents the industrial test case for the AGV system. Section 3 describes the holonic architecture of the system. Section 4 considers some use-cases showing how the holonic architecture can support the required flexibility. Finally, section 5 presents discussion and conclusion.

2 Industrial Application and Physical Devices

2.1 Industrial Background of the AGV System

The industrial background of the AGV system is two medium sized enterprises producing consumer goods. Their manufacturing process is a typical example of mass customization: they manufacture individually tailored products on a large scale.

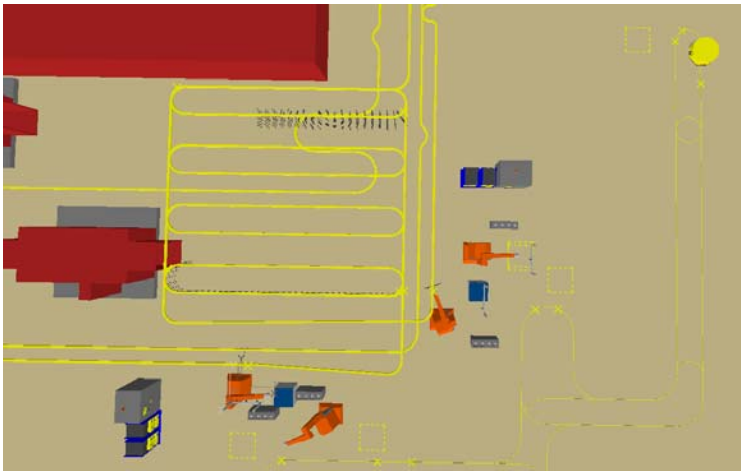


Fig. 1. The uploading(right side) and downloading(bottom side) cells of the paint department in QUEST

This paper is written as part of a larger project which focuses on the automation of the paint department and its surrounding logistic processes. In an earlier work the current paint process was analyzed and a new automated paint solution was proposed. The new automated paint-shop department and its logical control has been thoroughly documented. The result of this work serves as a reference system to be used for the further development of the proposed holonic AGV system.

Figure 1 shows the proposed system simulated in QUEST, a commercial simulation program. The heavy yellow wires are the tracks of the overhead conveyor which transports the painting carriers past the upload cell, the download cell, and through the paint process. All the red boxes make up the devices for the paint process, such as washing, drying, painting, hardening. The four robots are grouped in an upload cell and a download cell. In the upload cell one robot grips parts from bin and places them on an AGV, the other grips part from the AGV and hangs them on the conveyor belt. The download cell has not been setup yet but follows the same principles.

The role of the AGV system in the paint process is to perform the resupply of components for upload, and the transfer of downloaded components. The AGV system will also perform the transport tasks inside the cells, in-between the industrial robots, thus allowing for a better utilization of the shop-floor area e.g. by physically splitting the bin-picking operation from the uploading operation. For the industrial partners the AGV system is not viewed as a separate goal but rather as a support service for the manufacturing cells we are working on and eventually the rest of the manufacturing process.

A concrete challenge in this transport application is that it will not be possible to reserve pathways in all areas where the AGVs will be driving: There are no static paths nor static nodes and the pathways cannot be guaranteed to be free of obstacles. This uncontrollable environment requires advanced obstacle avoidance capabilities. This fact, as well as the importance of flexibility in this project, has driven our focus on free-roaming AGVs.

2.2 The Laboratory Demonstrator

To demonstrate the feasibility of the proposed paint process, and to experiment with new technologies, a laboratory demonstrator was setup. The functionality contained within the upload cell together with the corresponding resupply of components constitutes the basis for the laboratory demonstrator.

The implemented AGV part of the laboratory demonstrator consists of a working AGV that communicates with a vision system used for accurate positioning within the laboratory. The system is in the process of being extended: two AGVs are being assembled and several more are planned. In addition, several vision systems are being set up. By design, the AGV fleet will be heterogeneous to be able to experiment with AGV-dependent optimization for the manufacturing system.

The first test case is to transport components within the uploading cell. A bin-picker robot picks parts from a container and organise them on an AGV. When the task is accomplished the AGV drives to an upload robot where components are uploaded to the power-and-free conveyor to get painted (cf. Fig.2).

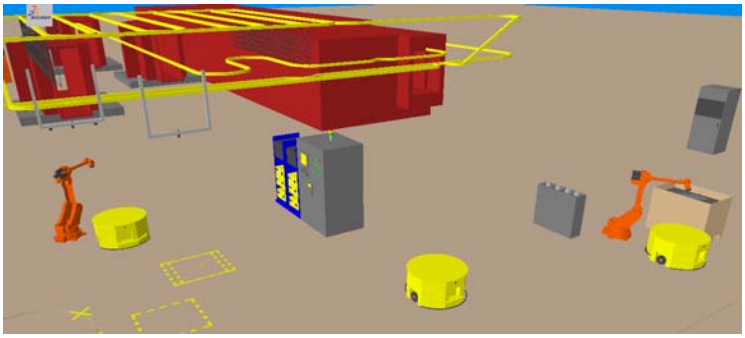


Fig. 2. The split upload cell using AGVs for transport between the two robots

3 The Holonic AGV System for Part Manufacturing

AGV systems are, by nature, distributed and flexible. When they, in addition, are composed of free-roaming AGVs, with local navigation ability, they match physically the definition of autonomous holons in the holonic manufacturing concept. As such we see holonic manufacturing as a natural choice for the control architecture of AGV systems.

Compared to a more traditional architecture the holonic architecture offers off-line and on-line flexibility that makes it possible to implement the required adaptability of the system, thus exploiting the possibilities offered by free roaming AGVs. The off-line structural flexibility of the holonic architecture enables the implementation of features of heterarchic and hierarchic organization[4]. In addition, holonic manufacturing has a built-in concept of on-line aggregation of holons into a new holon, that can be used to represent the physical, temporary association of holons to a specific task (cf. Sec.4).

The AGV system has been designed based on the published literature on holonic manufacturing and material handling system. The system is an implementation of the PROSA architecture[4] and is in several aspects inspired by the work done by S.C. Srivastavaet al[7] and Babiceanu et al[10].

Scalability in holonic material handling systems is a central issue which has been little mentioned in publications. In the design process of our proposed AGV system we have taken scalability issues into account from the beginning. We have used a distributed architecture and have avoided communication bottlenecks and central servers where practical.

3.1 The Holons

The proposed AGV system is composed of the four base holons defined in the PROSA architecture.

– Resource holons represent the physical devices. In our cases resource holons are AGVs or groups of AGVs.
– Product holons represent the part to manufacture. In our case they are containers for information about the parts to transport.
– Order holons are generated for each transport request. They are the driving force of the holonic system. They compete or collaborate with each other to allocate AGVs and process their transport need.
– Staff holons are adviser holons. They do not execute actions directly but give advices to other holons. A typical example is the scheduling holon which generates an 'optimal' schedule and advice it to other holons.

A simplified overview of the architecture is shown in Fig.3. Resource holons are represented in rectangles, order holons in ovals and staff holons in hexagons. It shows independent AGV holons and AGV holons aggregated in an AGV-group. It also shows one path-planning holon inside an AGV holon, although this is not a requirement: path-planning holons can be part of AGV holons or shared.

The main components of the AGV system have been split into autonomous entities: low-level AGV control, traffic, scheduling, path-planning, ordering, localisation. Not all components need to be implemented as holons. Entities can be implemented as separate processes, on servers or AGVs, but as soon as they can deliver useful information to other entities it makes sense to implement them as holons. Some entities, like ordering, are implemented using many order holons.

It is our experience that industrial partners often desire central algorithms for critical functions like scheduling. A possible implementation of such critical functions is to use a unique central holon or several holons using the same algorithm.

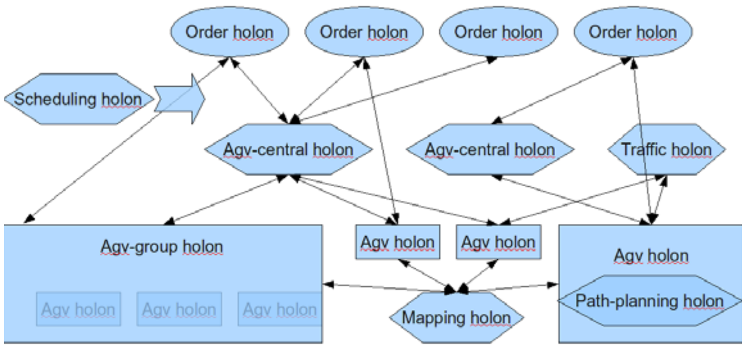


Fig. 3. Simplified overview of the AGV system control architecture

The algorithm must have a deterministic result so that holons eventually will end up with the same conclusions. This requires a system resilience to temporary disagreements. Resilience can be implemented in a holonic system by using staff holons giving advices, and not orders to other holons. In the disagreement period, the obtained result might not be optimal, but we think that the added flexibility and scalability of the solution overweight this issue.

A more detailed description of the main holon types is presented in the following subsections.

The AGV Holon. The AGV holon is the resource holon controlling the physical AGV and as such it runs directly on the physical AGV. In our system the AGVs are required to be able to run holons and communicate through a wireless network.

An AGV holon has a queue of order holon tasks which are periodically rescheduled. To support AGV-groups and special situations within the manufacturing cell AGV holons can enter a slave-state, thus letting another AGV holon or holon from the manufacturing take over the high level physical control of the AGV.

The Order Holon. The order holon is the specialized order holon for the AGV system. An order holon is created by a generic manufacturing order holon to solve a transport issue.

Order holons are the driving force of the holonic system. As long as its transport request is not executing, an order holon constantly queries the AGV-central holons for available AGVs and expected scheduling data. If it gets a better offer, or does not yet have an AGV assigned, it contacts the candidate AGV holon and request to be appended to its order queue.

The order holon also periodically contacts its assigned AGV to keep eachother aware of their current state. If an AGV holon does not get information from a order holon for a period of time, it removes it from its transport queue.

The AGV Central Holon. The AGV-central holon is a staff holon. Its name is inspired by an analogy to a ’taxi central’. The role of the AGV-central holons is to increase scalability by providing an interface to AGV holons. Each AGV uses one AGV-central holon chosen using, for example, geographic parameters.

AGV-central holons answer queries from order holons and advice them to an AGV. Order holons call several AGV-central holons and choose the best offer.

The Group Holon. Group holons are created when several AGVs link themselves together to a special task. The group holon is an abstract holon type inherited by specialized group holons cf. Sect.4.1. In some cases it may appear as one AGV holon to other holons.

The Scheduling Holon. The scheduling holon is a staff holon providing scheduling data to other holons. These holons are a necessary complement to local scheduling information from the AGV holon, since AGV holons have a limited vision of the global manufacturing system. The scheduling holons give advices to order holons or AGV-central holon to influence the choice of AGVs by order holons.

By communicating with other holons it keeps an up-to-date view of the manufacturing system, generates an 'optimal' scheduling and tries to get it applied by trying to re-affect order holons to AGVs.

The Path Planning Holon. The path-planning holon is a staff holon. It is a holon computing an optimal transport path for an AGV. It needs a map of the shop-floor obtained through the mapping holon and optional information from the online-traffic-control holon. Since the AGVs are capable of local navigation, a path is, in our context, defined as a list of points to be loosely reached. The path-planning holon can either be part of an AGV holon or be shared between several AGV holons.

The Mapping Holon. This is a staff holon that collects mapping information from all running AGVs in a geographic area. Mapping is a crucial feature of AGV systems for localization, navigation and path planning. Some of the AGVs are using Self Localization And Mapping algorithms, thus they depend on an up-to-date map to know their current position.

The idea is that all the AGVs participate in the on-line mapping of the shop-floor, thus reducing the need for manual updating of the current map. AGVs send periodically a report on their modification to their internal map, thus informing other AGVs of obstacles and changes on the shop-floor.

This system can used as the basis for the implementation of specialized scout AGVs that constantly investigate the shop-floor and maintain an up-to-date map. Unused AGVs could also be used to investigate unknown areas.

The Online Traffic Controller Holon. This is a staff holon whose role is to survey the AGV traffic in a geographical area and give advices to AGV holons to sort out conflicts and avoid congestions.

4 Holonic Use-Cases

This section details some use-cases as they have been defined for the AGV system. They focus on showing how the holonic architecture can be used to increase the flexibility of the AGV system.

4.1 AGV Holons as One AGV Holon

By this cryptic name we mean applying the holonic fractal concept to link together several AGVs to a special task. A group AGV can appear to the external holonic environment as a specialized AGV holon. The constituting AGV holons do not reply to requests as individuals but through the group holon. Examples of such applications are:

- Replace a palette conveyor with several AGVs: Several AGV holons enter a special 'pallet' state and one AGV holon start acting as a pallet conveyor supervisor. Required specialized holons are started. e.g. scheduling.
- Create a transport chain for an transport order with high priority.
- Transport large objects. The AGVs can link together their lower-level controllers thus allowing on AGV holon to control directly all physical AGVs. The AGVs can even be linked together mechanically.

4.2 AGV Holon as Part of a Manufacturing Cell Holon

A fundamental concept of our system is that AGVs are an integrated part of the manufacturing system. AGV holons can be assigned to a cell holon or, even, a robot holon for a given time.

- AGVs can be assigned to a robot system, thus allowing direct ordering from the robot to the AGV, effectively bypassing all scheduling and negotiations with other holons. It can be used, for example, to implement efficient cooperation between two robots out of each others reach: when a robot is finished placing parts on the AGV it communicates directly to the AGV to send it to next robot.
- In a similar idea AGVs can be used to increase the reach of an industrial robot. Thus allowing a smaller robot to temporary accomplish tasks usually requiring bigger robots.
- AGVs could also be used to push objects in a cell if required.
- AGVs can be affiliated to a robot on a track. An example taken from our paint conveyor: If instead of using a power-and-free conveyor for the hangers we would use a simple trolley conveyor and an industrial robot on rail. We could then follow the robot with the AGV while it is hanging parts on the conveyor, thus avoiding travel back and forth for the robot to pick parts. This solution would require a precise positioning system for the AGV or/and a vision system to give real-time part positions to the industrial robot.

4.3 The AGV as a Self Contained AGV System

In this concept a physical AGV represents the whole AGV System: the entire AGV system is started by starting one AGV. This feature is desired to simplify the administration of the system and to solve some specific scenarios e.g. transport to external storages areas outside the wireless network.

An AGV holon has a list of holons to connect, e.g. path-planning holon and AGV-central holons. Some of them are optional and others are required. If a required holon is not found then it is started either on the local AGV or on an available server.

A consequence of allowing holons to start other required holons, is that a mechanism is required to handle 'excessive' holons of one type. This is typical issue of distributed systems, holons of an excessive type need a consensus on who will close itself.

5 Discussion and Conclusion

In this paper we have presented the industrial background of an AGV system under development. The need for a flexible solution has been identified and a control architecture using the holonic manufacturing paradigm proposed. The paper has also presented, through several use-cases, how the holonic architecture can support the required flexibility of the AGV system.

In our proposed system there is clearly an emphases on the process viewpoint as defined in the reference PROSA architecture: It has been a conscious choice to focus on the AGV system as a support component of the manufacturing cell. As a consequence, the described use-cases are concerned with the interactions between the AGVs and the manufacturing cell and much less with high level scheduling and traffic management. One might say that AGV system presented only partially defines a material handling solution but we see this system as working subsystem from which a larger system can be composed, an important characteristic of holonic manufacturing systems[11].

6 Future Work

Implementation of the proposed holonic AGV system will continue during the next years in cooperation with the industrial partners. Much work has yet to be done to evaluate the performance, scalability and flexibility of the proposed holonic system and to demonstrate the industrial feasibility of the proposed use-cases.

Acknowledgements

This paper presents research results obtained through work in the IntelliFeed and CRI Norman projects sponsored by The Research Council of Norway.

References

1. Zipkin, P.: The Limits of Mass Customization. Harvard Business Review 75, 91–101 (1997)
2. Davis, S.: Future Perfect. Reading, Massachusetts (1987)
3. Van Brussel, H.: Holonic Manufacturing Systems The Vision Matching the Problem. In: Proceedings of the 1st European Conference on Holonic Manufacturing Systems, Hannover, Germany, IFW-Hannover (1994)
4. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems: PROSA. Computers in Industry 37(3), 255–274 (1998)
5. Leitão, P.: Agent-based distributed manufacturing control: A state-of-the-art survey. Engineering Applications of Artificial Intelligence (2008)
6. Liu, S., Gruver, W., Kotak, D., Bardi, S.: Holonic manufacturing system for distributed control of automated guided vehicles. In: 2000 IEEE International Conference on Systems, Man, and Cybernetics, vol. 3 (2000)
7. Srivastava, S., Choudhary, A., Kumar, S., Tiwari, M.: Development of an intelligent agent-based AGV controller for a flexible manufacturing system. The International Journal of Advanced Manufacturing Technology, 1–18 (2007)
8. Babiceanu, R.: Holonic-based control system for automated material handling systems. PhD thesis (2005)
9. Wallace, A.: Application of AI to AGV control-agent control of AGVs. International Journal of Production Research 39(4), 709–726 (2001)
10. Babiceanu, R., Chen, F., Sturges, R.: Framework for the control of automated material-handling systems using the holonic manufacturing approach. International Journal of Production Research 42(17), 3551–3564 (2004)
11. Valckenaers, P., Van Brussel, H.H.: Fundamental insights into holonic systems design. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS, vol. 3593, pp. 11–22. Springer, Heidelberg (2005)

Safety Discrete Event Models for Holonic Cyclic Manufacturing Systems

Calin Ciufudean[*] and Constantin Filote[*]

"Stefan cel Mare" University, University str. 9,
720225 Suceava, Romania
`{Calin Ciufudean,Constantin Filote}calin@eed.usv.ro, filote@eed.usv.ro`

Abstract. In this paper the expression "holonic cyclic manufacturing systems" refers to complex assembly/disassembly systems or fork/join systems, kanban systems, and in general, to any discrete event system that transforms raw material and/or components into products. Such a system is said to be cyclic if it provides the same sequence of products indefinitely. This paper considers the scheduling of holonic cyclic manufacturing systems and describes a new approach using Petri nets formalism. We propose an approach to frame the optimum schedule of holonic cyclic manufacturing systems in order to maximize the throughput while minimize the work in process. We also propose an algorithm to verify the optimum schedule.

Keywords: Petri net, event graph, holonic cyclic manufacturing system, minimum cyclic time, kanban system.

1 Introduction

The explosive development of information and communication technology networks (ICTNs) is designing and implementing more sophisticated and flexible approaches for measuring and controlling manufacturing processes. It does this by linking high level systems, such as enterprise research planning (ERP) and customer relation ship management (CRM) to low-level, real-time control devices such as digital signal processing (DSP) programmable logic controllers (PLCs), robots, conveyors, milling machines, etc [1-3]. For these approaches, the supervisory controllers can be distributed across several different computational devices. Following the holonic approach, the supervisory controller can be considered as a single holon associated with a physical device, although atomically it consists of several sub-holons, one for each computational device involved in the control of physical device [3]. For a whole-parted system Koestler denoted the term "holon", derived from the Greek word *holos* = whole with the suffix –on (like neutron or proton) pointing out the part characteristic [4]. A holon could compile strategy out of its rules, which fits to its intentions and

[*] Calin Ciufudean and Constantin Filote are with the Electrical Engineering and Computer Science Department of the "Stefan cel Mare" University.

goals and with the interpolation of the environment. Typical holonic applications or models are applied mostly to the production processes, called holonic manufacturing systems (HMSs); we notice that similar to these approaches, they are modeled domain specifically [5, 6]. A complex holonic structure has a hierarchical order, rules and strategies framed by communication and reaction abilities. Most of the time, the hierarchies are looked at as rigid and inflexible shapes. However, some approaches are not following this like the approaches described in [6], and the one proposed here. An interesting situation occurs in both designing and controlling flexible manufacturing systems (FMSs), where there is no hardware support to ensure synchronization of the common resources flows (e.g. practical situations where the outcome that changes the states of FMS depends on the global sequence of events). In order to model and to ensure an optimized control of these situations, we propose an approach based on fundamental class of Petri nets that are well known to produce safe models and safe protocols for common flows of resources in large and complex systems.

In this paper, we focus our discussion on techniques for the predicting and verifying the performance of holonic distributed systems. We consider a holonic distributed system as a loosely or a tightly coupled processing element working co-operatively and concurrently on a set of related tasks. In general, there are two approaches for performance evaluation [7]: deterministic models and probabilistic models. In deterministic models, it is usually assumed that the task arrival times, the task execution times, and the synchronization involved are known in advance to the analysis. This approach is very useful for performance evaluation of real-time control systems with hard deadline requirements. In probabilistic models, the task arrival rates and the task service time are usually specified by probabilistic distribution functions. Probabilistic models usually give a gross prediction on the performance of a system and are usually used for the early stages of system design when the system characteristics are not well understood. We analyze the performance of holonic distributed systems and, in order to model clearly the synchronization involved in concurrent systems, the Petri net model is chosen. In this paper we consider event graphs as Petri nets in which each place has one input transition and one output transition. It has been shown that distributed systems can be modeled as event graphs [8, 9]. When the manufacturing times are deterministic (respectively stochastic), the cycle time (respectively the mean cycle time) of the model is the period (respectively the mean period) required to manufacture a given set of parts which fits with the required ratios. The smaller the cycle time (respectively the mean cycle time) the higher the productivity of the system.

When the firing times of transitions are deterministic, it is possible to define the cycle time of an elementary circuit. This is given by the ratio between the sum of the firing times associated with the transitions of the circuit and the number of the tokens in the places of the circuit, which is constant (we consider strongly connected graphs with the number of tokens in any elementary circuit, constant):

$$C_i = \frac{T_i}{N_i} \tag{1}$$

Where i = 1, 2, ... , n number of elementary circuits of the graph; C_i = cycle time of elementary circuits *i*; $T_i = \sum_{i}^{n} r_i$ = sum of the execution times of the transition in circuit *i*; $N_i = \sum_{i}^{n} M_i$ = total number of tokens in the places in circuit *i*.

In this case it has been proven that the cycle time of a strongly connected event graph is equal to the greatest cycle time of all elementary circuits. Furthermore, given a value C* greater than the largest firing time of all transitions, an algorithm has been proposed in [10] to reach a cycle time less than C*, while minimizing a linear combination of the token number in the places. The coefficients of the linear combination are the elements of a p-invariant. When the event graph is the model of a ratio-driven distributed system (such as manufacturing system), C* has to be greater than the largest cycle time of all command circuits [11]. A command circuit is an elementary circuit, which joins the transition that models the operations performed on the same machine [23 - 25]. Such a circuit contains one token to prevent more than one transition firing at any time in each elementary circuit.

In other words, C* must be greater than the time required by the bottleneck machine to perform a sequence of parts which fits with the production ratios. In the case of random firing times, it is no longer suitable for the elementary circuits to evaluate the behavior of the event graph in order to reach a given performance. Thus, the results presented in this paper, which aim at reaching a given mean cycle time in a steady state while minimizing a linear combination of the place markings, are particularly important at the preliminary design level of manufacturing systems working on a ratio-driven basis.

We also consider a few techniques for scheduling and verifying the throughput of holonic cyclic manufacturing systems (HCMSs). A holonic cyclic production system manufactures a set of products at a constant frequency. We notice that any holonic cyclic production system can be modeled as an event graph, and therefore it is possible to drive the optimal throughput using the properties of the event graphs [12 - 14] by scheduling the tasks of the HCMSs in order to minimize the work in process. One may notice that nowadays often these production systems display a distributed configuration, e.g. they can be seen as distributed systems. Section 2 frames the optimum behavior of the Petri net model of a holonic cyclic manufacturing system. In section 3 we give a sufficient condition for optimality, e.g. an algorithm that maximizes the throughput of scheduling production lines and we propose an algorithm to evaluate the bounds used to calculate the average cycle time. Section 4 concludes the paper.

2 Framing the Mean Cycle Time

In this paper we examine the holonic cyclic manufacturing systems (HCMSs), e.g. production tasks including assembly/disassembly or fork/join ones, with unreliable machines displayed in Jakson (infinite - capacity) networks and we consider the problem of maximizing the throughput by achieving a balance among the processing and repair of all machines under certain economical criteria.

It has been proven [14] that in an event graph a marking belonging to the optimal solution under a periodic operational mode (POM) is an optimal solution under the earliest operational mode (EOM). So, we consider the earliest operational mode of the event graph, and we assume only non pre-emptive transition firings. We further assume that, when the transition fires, the related tokens remain in the input places until the firing process ends. They then disappear, and one new token appears in each output place of the transition. We use the following notations: M_i = the marking of the elementary circuits, $i \in N$; $X_t^k \in R^+$ = random variable generating the time required for the kth firing of the transition t, $k \in N$; $I_t(n)$ = instant of the nth firing initiation of the transition t; E = set of elementary circuits; s(e) = sum of the random variables generating the firing; $\sum_{t \in e} X_t^1$ = sum of times of the transitions belonging to e; E_t = set of elementary circuits containing the transition t.

We assume that the sequences of transition firing times are independent sequences of integrable random variables. It was proven in [15] that there is a positive constant $s(M_0)$ so that:

$$\lim_{n \to \infty} \frac{I_t(n)}{n} = C_m \tag{2}$$

Where: C_m = the average cycle time of the event graph.

Furthermore, we denote by m_t the mean value of X_t^k and by q_t the standard deviation of X_t^k, i.e., $m_t = F[X_t^k]$ and $q_t^2 = F[(X_t^k - m_t)^2]$.

2.1 The Lower Bound of the Mean Cycle Time

The cycle time of the deterministic problem obtained by replacing the random variables which generate the firing times by their mean values, is a lower bound of the mean cycle time [16]. The following relation proven in [17] provides a better lower bound for the value of the mean cycle time than the previous one:

$$C^{**} \geq \max_{e \in E} F[\max\{ \frac{s\lfloor e \setminus \{t^*(e)\} + m_{t^*(e)} \rfloor}{M_0(e)},\ m_{t^*(e)} \}] \tag{3}$$

Where $t^*(e)$ is a transition, belonging to event e, with the greatest average firing time, i.e., $m_{t^*(e)} = \max_{t \in e} m_t$, and $s[e \setminus \{t^*(e)\}]$ is the sum of the firing times of the transitions belonging to e except $t^*(e)$.

2.2 The Upper Bound of the Mean Cycle Time

With M_0 being the initial marking, we derive a marking M_1 from M_0 by leaving the places which are empty in M_0, empty in M_1, and by reducing to one the number of tokens in the places containing more than one token in M_0.

Thus, $M_1(p) \leq M_0(p)$ for any set of places of the strongly connected event graph. An earliest operation mode running with the initial marking M_1 leads to a greater mean cycle time than the one obtained when starting from M_0. Then, starting from M_1, we apply to the event graph the earliest operation mode, but we block the tokens as soon as they reach a place already marked in M_1. This operation mode is referred to as the constrained mode [18]. We denote by C* the mean cycle time obtained by using the constrained operation mode when M_1 is the initial marking. We know [8, 9] that C* is greater than the mean cycle time obtained by using the earliest operation mode starting from M_1 which, in turn, is greater than the mean cycle time obtained with the earliest operation mode when the initial marking is M_0. Thus, C* is an upper bound of the solution to our problem (i.e. the mean cycle time obtained starting from M_0 when using the earliest operation mode). The following relation defines this upper bound:

$$C^* = F[\max_{z \in Z} s(z)] \tag{4}$$

Where, Z is the set of directed paths verifying the following properties:

- the origin and the extremity of any path is a marked place;
- there is no marked place between the origin and the extremity of the path.

3 The Evaluation of the Optimum Schedule of a HCMS

The problem is to determine the time at which each task should begin during the cycle. We assume that the set of tasks does not change, the tasks are not preemptive, and the task processing is deterministic. We determine two bounds for the mean cycle time of a Petri net model of a HCMS. The next step is to give an algorithm for optimizing the Petri net model in order to determine a feasible scheduling for the production cycle and to minimize the work in process, e.g. the number of cycles necessary to complete one item of each product, given that schedule [19-22].

Algorithm:

1. The tasks of the HCMS that require resources are scheduled in any order. We notice by s_i and by e_i the start, respectively the end time of task *i*. All tasks start in the interval $[0, C^*]$, and the resources can perform maximum one task at a time, where C^* is given by relation (4).
2. For each product p = 1, 2, …, n go to step 3 for the first task and go to step 4 for the rest of the tasks.
3. Let *i* be the first task in the sequence of tasks T_k, k = 1, 2, …, m necessary to produce one unit of k. Assign the table $T_k(i) = 0$ for this task.
4. For each product label task $T_k(j)$ as follows: if $s_j > e_i$ then set $T_k(j) = T_k(j) + 1$, otherwise set $T_k(j) = T_k(i)$.
5. For each product calculate the production time as follows:

   $C_p = [T_p(l) - T_p(f)] \cdot C_{med} + s_l - s_f$, where f is the first task, and l is the last one, in the sequences of tasks T_k, and $C_{med} = (C^* + C^{**})/2$, with C^{**} and C^* given by relation (3) , respectively relation (4).

6. For the entire production the total average work in process is $\sum_{p=1}^{n} \frac{C_p}{C_{med}}$, where p=1, 2, …, n is the number of products in a complete production cycle.

In the reminder of the previous section, we compare the previous bounds with the existing ones. Under the assumption of non-preemptive transition firing, it was proven in [16] that:

$$C^{**} \geq \max_{e \in E} F[\max\{ \frac{s\lfloor e \setminus \{t^*(e)\} + m_{t^*(e)} \rfloor}{M_0(e)}, m_{t^*(e)} \}] \text{ = old lower bound} \tag{5}$$

$$C^{*} \leq \sum_{t} m_t \text{ = old upper bound} \tag{6}$$

The following relations show that the new bounds are better than the old ones. But how close are they to the optimal solution? In order to answer this question we give the next algorithm, inspired from the operational research area, for verifying the system's performance:

a) Express the token loading in a (p x p) matrix P, where p is the number of places in the Petri net model of the system. Entry (A, B) in the matrix equals x if there are x tokens in place A and place A is connected directly to place B by a transition; otherwise (A, B) equals 0.

b) Express the transition time in a (p x p) matrix Q. The entry (A, B) in the matrix equals to the mean values of the random variables which generate the firing times (i.e., X_t^k) if A is an input place of the transition "i" and B is its output place. Entry (A, B) contains the symbol "w" if A and B are not connected directly as described above.

c) Compute matrix $C_{med} \cdot P - Q$ (with $p - w = \infty$, and $C_{med} = (C^* + C^{**})/2$, for $p \in N$), than use Floyd's algorithm to compute the shortest distance between every pair of nodes using matrix CP-Q as the distance matrix. The result is stored in matrix S. There are three cases.

1) All diagonal entries of matrix S are positive (i.e., $C_{med} \cdot N_k - T_k > 0$ for all circuits - see relation (1)) the system performance is higher than the given requirement;
2) Some diagonal entries of matrix S are zero's and the rest are positive (i.e., $C_{med} \cdot N_k - T_k = 0$ for some circuits, and $C_{med} \cdot N_k - T_k > 0$ for the other circuits) - the system performance has just reached the given requirement;
3) Some diagonal entries of matrix S are negative (i.e., $C_{med} \cdot N_k - T_k < 0$ for some circuits) - the system performance is lower than the given requirement.

In addition we may say that when a decision-free system runs at its highest speed, $C_{med} \cdot N_k$ equals to T_k for the bottleneck circuit. This implies that the places in the bottleneck circuit will have zero diagonal entries in matrix S. The system performance can be improved by reducing the execution times of some transitions in the circuit, or introducing more concurrency in the circuit (by modifying the initial marking), or increasing the mean cycle time (by choosing another average value for it).

4 The Evaluation of Kanban Systems

As we well know, an event graph can be used to model a kanban system. An example of a simple production line will be used to exemplify the above discussed problems. The production line consists of two machining tools (M_1 and M_2), two robot arms and two conveyors. Each machining tool is serviced by a dedicated robot arm, which performs load and unload tasks. One conveyor is used to transport work-pieces, a maximum of two at a time. The other conveyor is used to transport empty pallets. There are three pallets available in the system. Each work-piece is processed on M_1 and M_2, in this order.

The stochastic timed Petri net model of this system is shown in Fig.1. The initial marking of the net is $(300012111)^T$. When the time delays are modeled as random variables, it has become a convention to associate the time delays with the transition only. The involved transition has the associated time delays expressed in time units. The random variables X_1, X_2, X_3, X_4 are assigned to the transitions t_1, t_2, t_3, t_4, respectively. X_1 is uniformly distributed on [0.2], X_2 and X_3 are random variables with $F[X_2] = 11$ t.u. and $F[X_3] = 1$ t.u. X_4 is a constant and equals to 17 t.u. The Petri net model contains four loops.

The time delays associated with these loops, as well as their token contents are:

1) loop: $t_1\, p_2\, t_2\, p_3\, t_3\, p_4\, t_4\, p_1\, t_1$, loop delay: 30 u.t., token sum: 3, cycle time: 10 u.t.
2) loop: $t_1\, p_2\, t_2\, p_5$ (or p_8) t_1, loop delay: 12 u.t., token sum: 1, cycle time: 12 u.t.
3) loop: $t_2\, p_3\, t_3\, p_6\, t_2$, loop delay: 2 u.t., token sum: 2, cycle time: 1 u.t.
4) loop: $t_3\, p_4\, t_4\, p_7$ (or p_9) t_3, loop delay: 18 u.t., token sum: 1, cycle time: 18 u.t.

Thus, the minimum cycle time is 18 time units. This means that it takes a minimum of 18 time units to transform a raw workpiece into a final product. Computing the lower-bound and the upper bound of the cycle time of the event graph from Fig.1. by using the relations (3) and (4) , we obtain the values: $C^{**} \geq 12$ u.t.; $C^{*} \leq 18$ u.t.

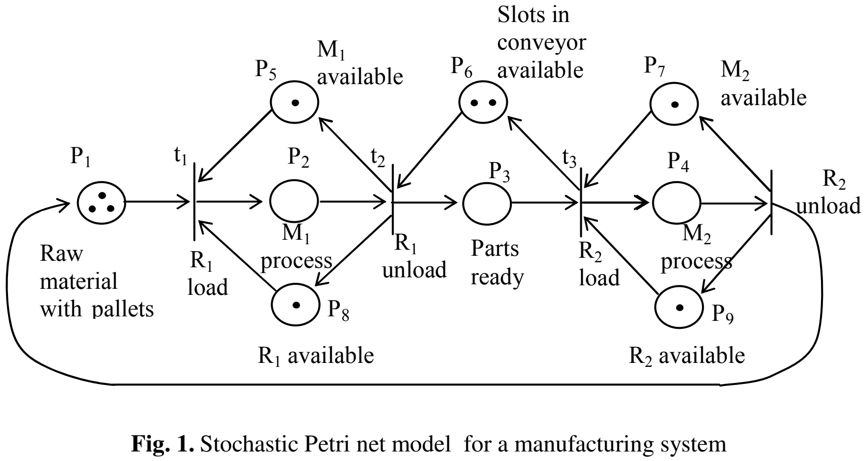


Fig. 1. Stochastic Petri net model for a manufacturing system

5 Conclusion

This work establishes a necessary relation between diagnosis and performances of holonic distributed systems, such as discrete event manufacturing systems. We have proposed a new architecture for the diagnosis and performance evaluation of holonic distributed systems using Petri nets formalisms and their properties [19, 20].

An important result in this paper is that it is always possible to reach a mean cycle time as close as possible to the greatest mean firing time using a finite marking, assuming that a transition cannot be fired by more than one token at each time. This result holds for any distribution of the transition firing time. An algorithm for verifying the distributed systems performance was introduced. An approach for computing the upper and lower bounds of the performance of a conservative general system is mentioned. However, the bounds produced may be loose.

Further research will focus on the conditions under which a mean cycle time can be reached with a finite marking.

Acknowledgments. The research was supported by Romanian National Council for Projects Management within Grant PN II No.71-065/2007.

References

1. Brusey, J., McFarlane, D.: Designing Communication Protocols for Holonic Control Devices using Elementary Nets. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS (LNAI), vol. 3593, pp. 76–86. Springer, Heidelberg (2005)
2. Peters, R., Tobben, H.: A Reference Model for Holonic Supply Chain Management. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS (LNAI), vol. 3593, pp. 221–232. Springer, Heidelberg (2005)
3. Bussman, S., McFarlane, D.C.: Rationales for Holonic Manufacturing Control. In: Proc. of the 2nd Int. Workshop on Intelligent Manufacturing Systems, Leuven, pp. 177–184 (1999)
4. Koestler, A.: Jenseits von Atomismus und Holismus – der Begriff des Holons, Wien (1970)
5. Delfmann, W., Albers, S.: Supply Chain Management in the Global Context, Koln (2000)
6. Lackmann, F., Nayabi, K., Hieber, R.: Supply Chain Management Software, Planungssysteme im Uberblick, Germany (2003)
7. Bacelli, F., Lin, Z.: Compresion properties of stochastic decision free Petri nets. IEEE Trans. Autom. Contr. 37(12), 1905–1920 (1992)
8. Laftit, S., Proth, J.M., Xie, X.L.: Optimization of invariant criteria for event graphs. IEEE Trans. Autom. Contr. 37(12), 547–555 (1992)
9. Proth, J.M., Xie, X.L.: Cycle time of stochastic event graphs: evaluation and marking optimization. IEEE Trans. Autom. Contr. 39(7), 1482–1486 (1994)
10. Campos, J., Chiola, G., Silva, M.: Properties and performance bounds for closed free choice synchronized monoclass quering networks. IEEE Trans. Autom. Contr. 36(12), 1368–1382 (1991)
11. Chauvert, F., Herrmann, J.W., Proth, J.M.: Optimization of Cyclic Production Sustems: A Heuristic Approach. IEEE Trans. On Robot. And Automat. 19(1), 15–154 (2003)
12. Proth, J.M., Xie, X.L.: Petri Nets. A Tool for Design and Management of Manufacturing Systems. Wiley, Chichester (1996)

13. Minis, I., Proth, J.M.: Production management in a Petri net environment. Recherche Operationelle 29(3), 321–352 (1995)
14. Zurawski, R., Zhon, M.C.: Petri nets and industrial applications: a tutorial. IEEE Trans. Ind. Electr. 41(6), 567–583 (1994)
15. Singh, J.: Operations Research. Penguin Books, Midlessex (1979)
16. Ciufudean, C., Popescu., D.: Scheduling Diagnosis of Flexible Manufacturing Systems. In: Proc. Int. Conf. on Autom, Contr. And Syst. Eng., Cairo (2005)
17. Ciufudean, C., Filote, C., Graur, A., Turcu, C.: Diagnosis of Complex Systems Using Ant Decision Petri Nets. In: The 1st Int. Conf. on Avail., Reliab. and Security, Vienna, pp. 473–482 (2006)
18. Ciufudean, C., Filote, C.: Performance Evaluation of Distributed Systems. In: International Conference on Control and Automation, Budapest, pp.21-25 (2005)
19. Yalcin, A., Boucher, T.O.: Deadlock Avoidance in Flexible Manufacturing Systems using Finite Automata. IEEE Trans. on Robot. And Automat. 16(4), 424–429 (2000)
20. Kouikoglou, V.: Optimal Rate Allocation in Unreliable Assembly/Disassembly Production Networks with Blocking. IEEE Trans. on Robot. And Automat. 16(4), 429–434 (2000)
21. Ciufudean, C., Satco, B., Filote, C.: Reliability Markov Chains for Security Data Transmitter Analysis. In: The 2nd Int. Conf. on Avail., Reliab. and Security, Vienna, pp. 886–892 (2007)
22. Recalde, L., Teruel, E., Silva, M.: Modeling and analysis of sequential processes that cooperate through buffers. IEEE Trans. on Rob. and Autom. 14(2), 267–277 (1998)
23. Zuo, M.J., Liu, B., Murthy, D.N.P.: Replacement-repair policy for multi-state deteriorating products under warranty. European Journal of Operational Research (123), 519–530 (2000)
24. Hopkins, M.: Strategies for determining causes of events, Technical Report R-306, UCLA Cognitive Systems Laboratory (2002)
25. Hall, K.H., Staron, R.J., Vrba, P.: Experience with Holonic and Agent-Based Control Systems and Their Adoption by Industry. In: Mařík, et al, pp. 1–10 (2005)

A Holonic Chain Conveyor Control System: An Application

Jan Van Belle, Bart Saint Germain, Paul Verstraete, Paul Valckenaers, Osman Ali, Hendrik Van Brussel, and Dirk Cattrysse

K.U. Leuven, Department of Mechanical Engineering,
Celestijnenlaan 300B, 3001 Heverlee, Belgium
jan.vanbelle@mech.kuleuven.be
http://www.mech.kuleuven.be

Abstract. The control of the flow of goods through an extensive chain conveyor system is a complex task. The currently used control system (based on dispatching rules) is robust but does not take advantage of all opportunities. An alternative approach makes use of a planning algorithm to determine the routing decisions. This requires however an ad hoc algorithm and regular maintenance. The paper examines how the concepts and principles of Holonic Manufacturing Execution Systems can be used to control the product flow. This holonic multi-agent approach makes the control system adaptive and reactive and requires less maintenance. To illustrate how disturbances are handled, the holonic approach is applied to a cross-dock distribution center equipped with chain conveyors.

Keywords: HMES, holonic, multi-agent, chain conveyor.

1 Introduction

Chain conveyors are often used for the internal transport of goods in factories, warehouses, etc. In many cases, several chains are connected to each other to form a complex transportation network. The control system of such a network decides about the routes that products follow and when these products are transported. Moreover, it must handle disturbances (e.g. a chain jam).

However, the control of the flow of goods through such a complex system is not straightforward. This is mainly caused by the non-linear nature of the underlying system, uncertainties and disturbances in the environment and the combinatorial growth of the decision space. Similar concerns have been addressed by the Holonic Manufacturing Execution System (HMES) described by Valckenaers [1]. This paper describes how the concepts and principles used in the HMES can be applied to control a chain conveyor system. As an industrial application, a cross-dock distribution center equipped with chain conveyors is considered.

After this introductory section, Sect. 2 describes chain conveyor systems in detail and explains the challenges to control these systems. Section 3 gives a brief overview of the Holonic Manufacturing Execution System. In Sect. 4, the current control practice is discussed. Also, an alternative approach based on a planning algorithm is described and the holonic approach is introduced. To clarify the holonic approach, the outcome of a case study of a cross-dock distribution center is presented in Sect. 5. The conclusions are summarized in Sect. 6.

2 Chain Conveyor System

A chain conveyor is a type of conveyor that consists of a chain driven by a motor. The chain forms a loop and moves continuously, usually at a low speed, to convey products. Different types of conveyor chains are common in industry. This paper considers an in-floor chain conveyor system; the developed concepts and principles can however also be applied to other types.

The most important components of an in-floor chain conveyor system are:[1]

Transport chain. A transport chain makes the carriers circulate continuously. The chain moves in a U-shaped channel that is completely anchored into the floor. At regular distances, it contains special links – so called 'pusher dogs' – that push the pins of the carriers forward. The chain moves at a constant speed, normally slow enough to allow staff to safely put carriers on and take carriers off the chain. The chain always forms a closed loop and has its own drive. It can be switched on and of when required (e.g. during breaks).

Transfer. A transfer is very similar to a transport chain, except that the loop is formed in the vertical plane. Hence, carriers can only move along half of the loop. A transfer is used to connect different chains to each other.

Carrier. Different types of carriers are commonly used, e.g. roll containers, universal product carriers or hand pallet trucks. Each carrier has a pin that can be attached to the chain in order to be be pushed forward. Typically, the carriers are equipped with a bar code or a RFID tag so that each carrier has a unique (electronic) ID.

Diverter. A diverter is used to switch a carrier from a chain to a transfer. When a diverter is enabled and a carrier passes by, the pin of the carrier is lifted up and moved above the connected transfer. When the next 'pusher dog' of that transfer arrives at the diverter, this pin is released and the carrier is pushed forward by the transfer.

Accumulation stop. When a transfer connects to a chain, the carriers moving on that transfer have to be placed on the chain at the correct time (i.e. the moment a free 'pusher dog' passes by) to avoid a collision. Therefore, at the end of a transfer, one or more accumulation stops are installed. The accumulation stop is responsible for holding the carrier for a certain time and releasing it at the correct moment. This is realized by lifting the pin of the carrier up and releasing it at the correct moment. Accumulation stops can also buffer carriers in a waiting line.

[1] One typical hardware implementation is described here, but other implementations are also possible.

Sensor. Different types of sensors are used to provide an accurate overview of the situation. At certain places ID readers can be installed. These are able to read out the bar code or the RFID tag of the passing carrier. Typically, the ID readers are installed at a small distance before each diverter. Based on the input from the sensors, the control system can decide whether or not the next diverter has to be enabled. Another type of sensor is a carrier detector. It is capable of detecting whether a 'pusher dog' pushes a carrier or not. This kind of sensors are, for instance, installed at a small distance before a chain merges with a transfer. The control system typically uses the information of the carrier detector to decide if an accumulation stop at the end of the transfer can release a carrier or not.

Control system. A central or distributed control system is responsible for operating and monitoring the chain conveyor system. It can actuate the different diverters and accumulation stops to control the flow of goods. The information of the sensors is used to make these decisions. The control system also has information regarding the state of the system and is able to track and trace the products.

By combining the above-mentioned components, a complex network of chain conveyors can be formed. Such a network can be used for the internal transport of goods, for instance in warehouses, cross-docking distribution centers or factories. Usually, 'operations' have to be performed on the products (e.g. manufacturing operations or value-added services like labeling and packaging). A chain conveyor system can be used to transport the goods between the different 'workstations' that execute the operations.

The control system is responsible for the management of the chain conveyor network. It must guarantee that the *necessary performance* is obtained to fulfill the predefined requirements (e.g. all trucks have to be loaded at the end of the day, the workstations always need to be supplied on time). The control of a large and complex network is however a hard job. There are *many decisions* that have to be taken (the state of accumulation stops and diverters, the speed of the chains and transfers, ...) to control the flow of the goods. These decisions have to be made based on *incomplete information*, e.g. the (exact) position of a carrier is only known when it is detected by a sensor. The environment in which chain conveyor systems are generally used - like the distribution and logistics sector - is subject to *uncertainties and disturbances* (e.g. variable operation times, equipment failures, delays, etc.). Also, decisions can have *long term repercussions*. For instance, when a carrier has to be transported to a heavy loaded part of the network, the control system can decide not to route the carrier to that part but to keep it on its current chain. In that case, the carrier has to make (at least) one extra loop before it can be routed to its destination. Because the time to make a complete loop can be significant, the consequences of such decisions have a serious impact on transportation time. Other carriers are also influenced because the load of the chain is increased.

3 Holonic Manufacturing Execution System

In this section the Holonic Manufacturing Execution System (HMES) is briefly described. An introduction to the HMES was given by Valckenaers in [1]. A detailed description of its software architecture can be found in [2].

The HMES is implemented with multi-agent technology as an instantiation of the PROSA reference architecture [3]. The acronym PROSA denotes *Product-Resource-Order-Staff Architecture* and refers to the composing types of holons. Each holon has a corresponding type of agent. A *resource agent* corresponds to a resource in the underlying system. It is responsible for managing this resource and for reflecting it to the remainder of the multi-agent system. A *product agent* corresponds to a product type. It contains the knowledge on how instances of its type can be produced by the factory resources (but has no knowledge about these created instances). An *order agent* corresponds to a task that needs to be executed. It handles the required resource allocations to get its product instance produced. Therefore, the order agent consults the product agent to find out what operation it should perform next and searches for the proper resources to accomplish it.

The coordination between the agents is inspired by natural systems, namely the coordination and control mechanisms of ant colonies [1]. In an ant colony, ants deposit information (pheromones) in their environment informing other ants about remote facts (e.g. how to find food). The PROSA agents create lightweight agents - called ant agents - for similar purposes (i.e. system-wide coordination). Basically, there are two types of ant agents: exploring ants and intention ants. Every order agent creates at regular time intervals *exploring ants* that scout for possible solutions. The different solutions reported by exploring ants are evaluated by the order agent and the most attractive solution is chosen to become its intention. Then, the order agent sends an *intention ant* to reserve the necessary capacity on the resources that the chosen solution comprises. Note that exploring ants are created regularly, even after the order agent has chosen its intention. In this way, the order agent can adapt to disturbances (e.g. the breakdown of a machine) or new opportunities (e.g. a repaired machine is available again).

The basic holons reflect a part of reality. They have to be able to answer what-if questions about their corresponding reality. For instance, when an exploring ant searches for a solution, it asks the resource holon *what* will be its completion time *if* it arrives at a certain time. To be able to answer this question, the resource holon has to know the dynamic behavior of its corresponding resource. Therefore, every holon contains a model of its corresponding reality. To model the dynamic behavior, multimodels are used [4,5]. Multimodels allow using different models with different formalisms and different levels of detail.

4 Control of a Chain Conveyor System

As explained in Sect. 2, the control of a chain conveyor system is a difficult task. Sections 4.1, 4.2 and 4.3 respectively describe the current approach, possible improvements by a planning algorithm and the holonic multi-agent approach.

4.1 Current Situation: Static Routing

Currently, a chain conveyor system is controlled in a static way. Each diverter has a routing table that defines for each product type if it has to take the carrier of the chain or let it pass. So all products from the same category follow the same route through the system. The routing tables are made manually and are only adapted when serious changes happen, for instance when the product mix changes seriously. The accumulation stops at the end of a transfer are only used to synchronize with the connected chain. When an accumulation stop holds a carrier, the carrier is released as soon there is a free 'pusher dog' of the chain available. The inflow on the chain conveyor system is controlled by a dispatching rule. When the chains are heavily loaded (e.g. 90 % of all 'pusher dogs' are occupied), the inflow is stopped until the load is decreased with a certain amount.

The current approach is *myopic*. The control system has no idea about the consequences of, for instance, the decision of a diverter on other parts of the chain conveyor system. However, this decision can have long term repercussions. Because of this myopia, the control system is *not capable of taking advantage of all possible opportunities*. E.g. it could be better to adapt the routing tables according to the current load of the conveyor chains and the products that have to be transported in the next time period. Also, when the chains are heavily loaded, it can be better to halt only the release of the products with the later due dates instead of stopping the inflow of all goods. This myopia makes it also *difficult to handle uncertainties and disturbances adequately*. For example, a broken down chain conveyor blocks all products that have to be routed along this chain even if a detour is possible. On the other hand, the used dispatching rules can be applied in (almost) all circumstances. Moreover, these rules are not vulnerable to errors in the known information (e.g. the current situation of the system, expected orders and deliveries, . . .), simply because they ignore most of this information. This makes the current approach *robust*. Other (less myopic) dispatching rules can also be used. However, each chain conveyor system requires *ad hoc dispatching rules* and *intensive maintenance* of these rules if the situation changes (e.g. a different product mix or another layout).

4.2 Dynamic Routing

An alternative approach tackles the static routing tables of the diverters by fixed routing tables. The diverters still use routing tables, but these tables are updated at regular times by a planning algorithm. This algorithm takes more elements into account, for instance, the goods that have to be processed during the next time interval, the due dates of the different products, etc. These dynamic routing tables can be changed frequently, depending on, among other things, the performance of the planning algorithm.

This approach takes more information into account and is expected to yield better results, depending on the chosen time interval and the planning algorithm. Assuming that the algorithm can recalculate quickly, the dynamic routing is *reactive* and can *handle uncertainties and disturbances adequately*. In comparison with the static routing, this approach is *less robust* because the algorithm makes use of information that is possibly incomplete or even erroneous. The updating of the routing tables can be automated by the planning algorithm. However, maintenance is still required. The algorithm itself needs *ongoing maintenance* to achieve the necessary performance. Another drawback of this approach is that each chain conveyor system requires an *ad hoc algorithm.*

4.3 Holonic Chain Conveyor Control System

A third approach consists in applying the concepts and principles of the Holonic Manufacturing Execution System (Sect. 3). The carriers are represented by order agents that send out exploring ants to search for solutions. The different chain conveyors, diverters, accumulation stops, etc. are represented by resource agents where the intention ants can reserve capacity for the order agent.

As opposed to the currently used approach, the holonic control system is *not myopic.* The intentions of the different carriers are propagated through the system as short-term forecasts and so the consequences of decisions become visible. One of the advantages of the Holonic Manufacturing Execution System is its *adaptability and reactivity.* Because the order agent keeps sending exploring and intention ants, also after it has declared its intention, it discovers changes and disturbances quickly. The order agent also maintains up-to-date alternative solutions and can switch to one of them if its current intention becomes infeasible. Contrary to the other approaches, this approach doesn't require an ad hoc solution for each chain conveyor system. Furthermore, *less maintenance* is needed to operate this control system to obtain the necessary performance.

5 Application: A Cross-Dock Distribution Center

To validate the proposed holonic approach, an implementation of the holonic chain conveyor control system is applied (in simulation) to a cross-dock distribution center that uses a chain conveyor system. The layout and components of the considered distribution center are described in Sect. 5.1. Section 5.2 explains multimodels and how they are used in the holonic control system. A detailed description of the behavior of the holonic chain conveyor control system in case of a disturbance is given in Sect. 5.3.

5.1 Cross-Dock Distribution Center

The considered distribution center (see Fig. 1) is a simplified version of a real-world case study. Trucks that arrive at the incoming docks ($In1$ and $In2$) are unloaded and the goods are placed on carriers. These carriers are then put (manually) on a conveyor chain ($Chain1$). This chain conveyor has a length of ca. 83 m and has 10 'pusher dogs' equally distributed along the length of the chain. Three transfers are connected to this chain ($Transfer1$, $Transfer2$ and $Transfer3$) by means of a diverter. Before each diverter, a sensor reads out the

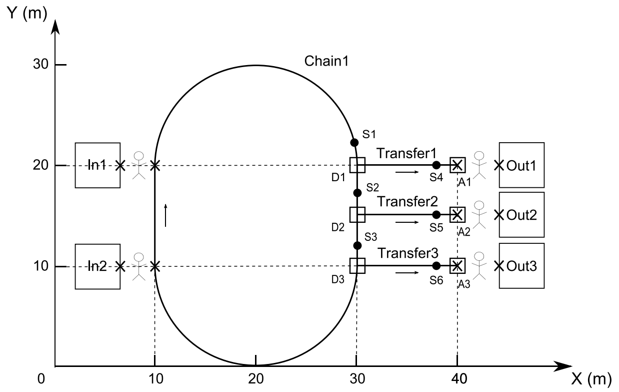


Fig. 1. Layout of the cross-dock distribution center

tag of the passing carrier. At the end of each transfer there is an accumulation stop with a carrier detector in front of it. These accumulation stops hold the carriers until they are removed by a worker and the goods are loaded in a truck at the outgoing docks (*Out*1, *Out*2 and *Out*3). The chain as well as the transfers move at a constant speed (1 m/s).

5.2 Multimodels

As explained in Sect. 3, each holon contains a model of its reality. The corresponding agent uses this model to answer what-if questions. The resource agent of a chain, for instance, uses this model to determine what the transportation time of a carrier is. This duration depends on the starting place and the destination of the carrier among other things. For the considered cross-dock distribution center, models of the behavior of the chain, the transfers, the diverters, the accumulation stops, the workers and the carriers are required.

The approach uses multimodels to model the behavior of the different components [4,5]. The Petri net formalism is used for the top-level of the multimodels. However, a transition can represent a sub-model and can have an 'in' and an 'out' boundary condition. These boundary conditions define when the sub-model is valid. An 'in' boundary condition determines when the sub-model is active and can be executed. The 'out' boundary condition defines when the sub-model is ready and becomes inactive again. A transition may fire when there is a token in all input places *and* the 'in' boundary condition is true. Then, the corresponding sub-model is executed until the 'out' boundary condition is true and a token is placed on all output places.

Figure 2 shows the top-level model that the diverters use. The model consists of four places and five transitions (four of them have a sub-model). When this model is executed, transition $T1$ can fire because the place *Start* contains a token and no 'in' boundary condition is defined. The sub-model $Waiting$ is then executed until the *Enabled* boundary condition is true. This sub-model represents the behavior of the diverter when it is idle. The 'out' boundary condition is

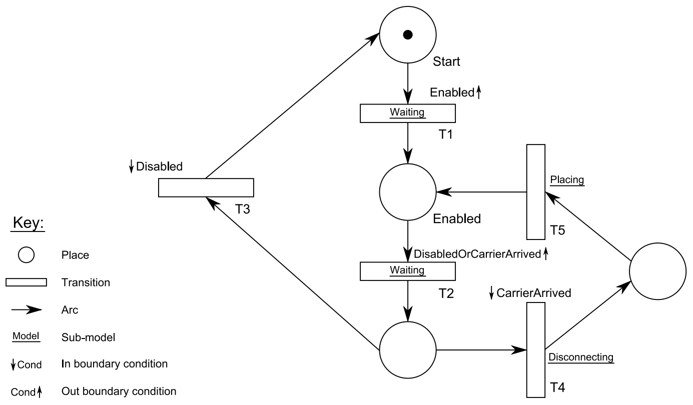


Fig. 2. Top-level model of a diverter

fulfilled when the diverter is 'enabled' and has to switch a carrier from its current chain to another one. Next, a token is placed in *Enabled* and transition $T2$ can fire. The corresponding sub-model – again *Waiting* – is executed until its 'out' boundary condition is true. This condition is true when the diverter is 'disabled' or a carrier arrives at the diverter. If *Disabled* becomes true, transition $T3$ fires and a token is placed on *Start*. If *CarrierArrived* is true, transition $T4$ and $T5$ are sequentially fired. First, *Disconnecting* is executed. This *Disconnecting* model represents the disconnecting of the carrier from its current chain. Subsequently, the carrier is placed on the other chain in the sub-model *Placing*. Afterwards, a token is placed in *Enabled* and *Waiting* is again executed until the diverter is 'disabled' or a next carrier arrives.

5.3 Simulation

In this section, the simulation results are described qualitatively. The simulation is performed on a self-developed simulation platform that applies software-in-the-loop simulation mode [5]. The focus of the simulation is on showing how the holonic chain conveyor control system reacts to a specific disturbance, the late arrival of a truck. Two scenarios are simulated. In the first scenario there are no disturbances, in the second scenario one of the trucks is later than expected.

In **scenario** 1, five trucks arrive at time 0, one at each dock. The two trucks at the incoming docks, which have to be unloaded, each contain 12 pallets. The three trucks at the outgoing docks are empty and have to be loaded with eight pallets, four pallets of both of the incoming trucks. The carriers can transport one pallet.

The outcome of the simulation shows that the pallets for the different outgoing trucks are mixed up when they are transported by *Chain*1. The simulation also reveals that the last pallet arrives at its destination after ca. 330 s.

The behavior of the holonic control system during the simulation is the following. When the incoming trucks arrive, an order agent is created for each pallet.

These order agents start sending out exploring ant agents immediately to find possible solutions. After the order agents have collected different solutions, they choose one of these solutions as their intention and send an intention ant to reserve the necessary capacity on the different resources (e.g. the use of a 'pusher dog' on a chain conveyor). The following example explains the behavior of the exploring ants. The order agent responsible for a pallet that has to be transported from $In1$ to $Out3$ creates an exploring ant that looks for a solution starting at time t_0. This exploring ant first visits the resource agent corresponding to the worker at dock $In1$ and asks if it is possible to be moved to $Chain1$ at time t_0. The resource agent responds with the arrival time t_1 if it is possible. Next, the exploring ant goes to the resource agent corresponding to $Chain1$ and asks to be conveyed to diverter $D3$ starting at t_1. Again, the resource agent checks if it can process the order and returns the end time t_2 of the operation. Subsequently, the exploring ant visits diverter $D3$, $Transfer3$, accumulation stop $A3$ and the worker at dock $Out3$. The different resource agents are capable of answering these kind of what-if questions by means of their multimodel that captures the behavior of the corresponding resources. When an order agent books its intention, it is possible that this booking does not succeed. This happens for instance when another order agent booked an overlapping solution just before. In this case, the order agent has to select another intention and try again to book it. In this way it can take the order agent several attempts to book an intention.

Scenario 2 is similar to scenario 1, but now one of the outgoing trucks arrives later than expected. The truck at dock $Out3$ arrives with a delay of 300 s.[2]

The outcome of the simulation shows that the inflow of the pallets is limited; the pallets for dock $Out3$ are not released immediately. In the beginning of the simulation only the pallets for $Out1$ and $Out2$ are transported. The last of these pallets reaches its destination already after ca. 210 s. The pallets that have $Out3$ as destination are released after 250 s. So they arrive at $Out3$ after 300 s, when the truck has already arrived. The last pallet reaches its destination after 410 s.

By controlling the inflow of the carriers, the holonic control system reduces the load of the chain conveyors. This desired effect is due to the behavior of the exploring ants when they find out that a resource is not available. To explain this behavior, the example of the exploring ant given for scenario 1 is resumed. When the exploring ant now arrives at $Out3$, the corresponding resource agent will inform the ant that the truck is not available until time 300 s. The exploring ant will propagate this information (about the unavailability) back along the path it has followed. The information is corrected for each successive resource to reflect the time interval in which it has no sense to arrive. The exploring ant goes back to the worker at $Out3$, then to the accumulation stop $A3$, to $Transfer3$ and so on. The ant returns until it arrives at a resource where the order can be stored for some time (in this case at the entrance of the system, $In1$).[3] The next

[2] This delay looks not large, but since the time needed to transport a pallet from $In2$ to $Out3$ is in normal circumstances only ca. 80 s, this delay is significant.

[3] If it is allowed to use $Chain1$ to store carriers, the exploring ant should only return up to $Chain1$. Carriers can be stored on a chain conveyor by making multiple loops.

exploring ants starting at $In1$ see this information and decide (probabilistically) to start at a later time so that they will not be blocked at that resource.

6 Conclusion

Chain conveyors are used in industry for the internal transport of products. In some cases a complex network of chains is formed. The control system of a chain network decides about the routing of the products and handles disturbances. The control of the flow of goods through such a complex system is not a straightforward task. The currently used control system is based on dispatching rules. This is a robust but myopic approach. The control system is not capable of taking advantage of all possible opportunities.

This paper proposes a holonic approach for the control of a chain conveyor system by applying the principles and concepts of a Holonic Manufacturing Execution System. This makes the control system adaptive and reactive. This approach is generically applicable and requires less maintenance. To illustrate how this approach can be applied and what benefits it can bring, a cross-dock distribution center equipped with chain conveyors is simulated. The case study reveals that this holonic approach is suitable to control the flow of goods and to deal with disturbances. Since the principles of the HMES are already successfully applied to different applications, it is expected that this approach is also appropriate for other chain conveyor systems.

Acknowledgments. This paper presents work funded by the Research Fund of the K.U. Leuven (Concerted Research Action on Autonomic Computing for Distributed Production Systems).

References

1. Valckenaers, P., Van Brussel, H.: Holonic manufacturing execution systems. CIRP Annals - Manufacturing Technology 54(1), 427–432 (2005)
2. Verstraete, P., Saint Germain, B., Valckenaers, P., Van Brussel, H., Van Belle, J., Hadeli: Engineering manufacturing control systems using PROSA and delegate MAS. International Journal of Agent-Oriented Software Engineering 2(1), 62–89 (2008)
3. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems: PROSA. Computers in Industry 37(3), 255–274 (1998)
4. Fishwick, P.A., Zeigler, B.P.: A multimodel methodology for qualitative model engineering. ACM Transactions on Modeling and Computer Simulation (TOMACS) 2(1), 52–81 (1992)
5. Verstraete, P.: Integrating existing scheduling techniques into the Holonic Manufacturing Execution System. PhD thesis, Katholieke Universiteit Leuven (2009)

A Multiagent System for Self-organisation of an 802.11 Mesh Network

John Debenham and Ante Prodan

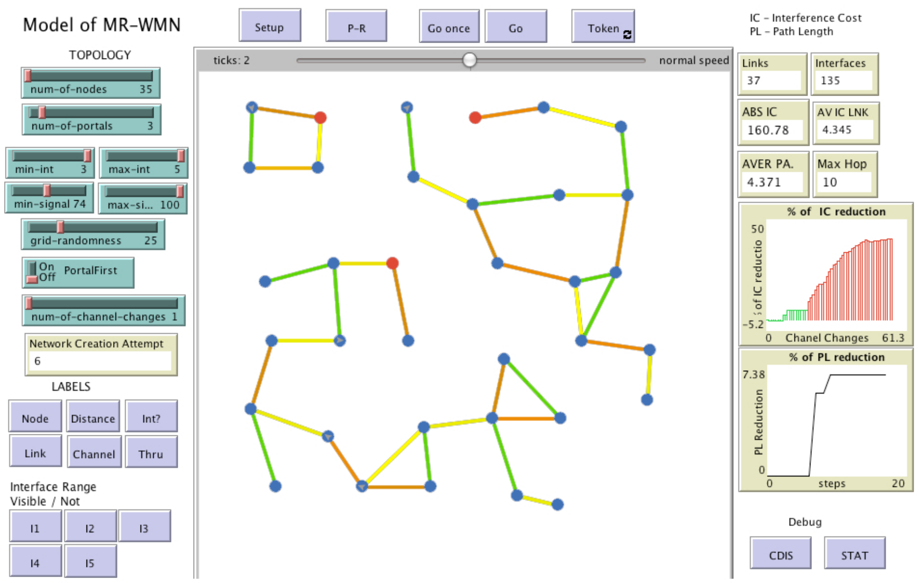
Centre for Quantum Computation & Intelligent Systems,
University of Technology, Sydney, Australia
{debenham,aprodan}@it.uts.edu.au

Abstract. The self-organisation of telecommunications networks has to confront the two challenges of the scalability and the stability of the solution. This paper describes a distributed, co-operative multiagent system in which agents make decisions based only on local knowledge — that guarantees scalability. Extensive simulations indicate that stability is ensured by the agent's making improvements to the network settings that improve the social performance for all agents in a two-hop range. Our overall goal is simply to reduce maintenance costs for such networks by removing the need for humans to tune the network settings.

1 Introduction

Recent work on 802.11 Mesh Networks, such as [1], is predicated on a network whose prime purpose is to route traffic to and from nodes connected to the wired network — in which case there is assumed to be no traffic between end-user nodes. This introduces the conceptual simplification that mesh nodes can be seen as being grouped into clusters around a wired node where each cluster has a tree-like structure, rooted at a wired node, that supports the traffic. This is the prime purpose of 802.11 Mesh Networks in practice. Where possible, we move away from any assumptions concerning tree-like structures with the aim of designing algorithms for the more general classes of "wireless ad-hoc networks" or "wireless mesh networks". This paper is based on previous work in the area of mesh networking, and in particular in distributed algorithms at Columbia University, Microsoft Research, University of Maryland and Georgia Institute of Technology. See also: [2], [3], [4] and [5].

There are three principal inputs that we assume are available to the methods described: a load model, a load-balancing algorithm and an interference model. The work described below makes no restrictions on these three inputs other than that they are available to every node in the mesh. The load model, and so too the load balancing algorithm, will only be of value to a method for self-organisation if together they enable future load to be predicted with some certainty. We assume that the load is predictable. We assume that if the external demands on a set of nodes S are known and that there is a *load balancing algorithm* — that may or may not be intelligent — that determines how the load is routed through S. We assume that the load balancing algorithm will determine how the load is allocated to each link in the mesh.


Model of MR-WMN

TOPOLOGY

num-of-nodes 35
num-of-portals 3
min-int 3
max-int 5
min-signal 74
max-si... 100
grid-randomness 25
On Off PortalFirst
num-of-channel-changes 1

Network Creation Attempt
6

ticks: 2 normal speed

IC – Interference Cost
PL – Path Length

Links	Interfaces
37	135
ABS IC	AV IC LNK
160.78	4.345
AVER PA.	Max Hop
4.371	10

Fig. 1. The implementation of the algorithms

The multiagent system described in the paper has been simulated; the simulation is available[1] on the World Wide Web. A screen shot is shown in Figure 1. The "Setup" button establishes a random topology in line with the TOPOLOGY settings on the left side. The "P-R" button reduces the path length, and the "Go" button reduces the interference cost using the algorithms described in this paper. The colours on the arcs denote which of the eleven 802.11 channels is used. The work has been conducted in collaboration with Alcatel-Lucent, Bell Labs, Paris.

The measurement of interference cost is discussed in Section 2. Methods for the adjusting the channels in a multi-radio mesh networks for predictable load are described in Section 3, and for adjusting the links in Section 4. Future plans are described in Section 5.

2 Measuring Interference Cost

Suppose that during some time interval Δt two interfaces a and b are transmitting and receiving on channels Γ_a and Γ_b. During Δt, the *interference limit* that interface x imposes on interface y, $\tau_{y|x}$, is a ratio being the loss of traffic volume that interface y could receive if interface x were to transmit persistently divided by the volume of traffic that interface y could receive if interface x was silent:

$$\tau_{y|x} = \frac{(m_y \mid \text{ interface } x \text{ silent}) - (m_y \mid \text{ interface } x \text{ persistent})}{m_y \mid \text{ interface } x \text{ silent}}$$

[1] http://www-staff.it.uts.edu.au/~debenham/homepage/holomas/

where m_y is the mean SNIR observed by interface y whilst listening on channel Γ_y, where as many measurements are made as is expedient in the calculation of this mean[2]. The *interference load* of each interface, v_a and v_b, is measured as a proportion, or percentage, of some time interval during which that interface is transmitting. Then the *observed interference* caused by interface b transmitting on channel Γ_b as experienced by interface a listening on channel Γ_a is: $\tau_{a|b} \times v_b$, and the *observed interference cost* to interface a is[3]:

$$f(a \mid b) \triangleq \tau_{a|b} \times v_b \times (1 - v_a)$$

and so to interface b:

$$f(b \mid a) = \tau_{b|a} \times v_a \times (1 - v_b)$$

Now consider the interference between one interface a and two other interfaces c and d. Following the argument above, the *observed interference* caused by interfaces c and d as experienced by interface a is[4]: $\tau_{a|c} \times v_c + \tau_{a|d} \times v_d - \tau_{a|\{c,d\}} \times v_c \times v_d$. The observed interference cost to interface a is:

$$f(a \mid \{c,d\}) = (1 - v_a) \times \left(\tau_{a|c} \times v_c + \tau_{a|d} \times v_d - \tau_{a|\{c,d\}} \times v_c \times v_d\right)$$

If interfaces at agents c and d are linked then they will transmit on the same channel Γ_β, and we ignore the possibility of them both transmitting at the same time[5]. Further suppose that v_β is the proportion of Δt for which either interface c or interface d is transmitting. Then for some κ_β, $0 \leq \kappa_\beta \leq 1$: $v_c = \kappa_\beta \times v_\beta$, and $v_d = (1 - \kappa_\beta) \times v_\beta$. Thus:

$$f(a \mid \beta) = (1 - v_a) \times v_\beta \times \left(\tau_{a|c} \times \kappa_\beta + \tau_{a|d} \times (1 - \kappa_\beta)\right)$$

Now suppose that interfaces a and b are linked, and that v_α is the proportion of Δt for which either interface a or interface b is transmitting. Then for some κ_α, $0 \leq \kappa_\alpha \leq 1$: $v_a = \kappa_\alpha \times v_\alpha$, $v_b = (1 - \kappa_\alpha) \times v_\alpha$. Then as a will only receive interference when it is listening to b transmitting:

$$f(a \mid \beta) = v_b \times v_\beta \times \left(\tau_{a|c} \times \kappa_\beta + \tau_{a|d} \times (1 - \kappa_\beta)\right)$$

and so:

$$\begin{aligned} f(\alpha \mid \beta) = (1 - \kappa_\alpha) \times v_\alpha \times v_\beta \times \left(\tau_{a|c} \times \kappa_\beta + \tau_{a|d} \times (1 - \kappa_\beta)\right) \\ + \kappa_\alpha \times v_\alpha \times v_\beta \times \left(\tau_{b|c} \times \kappa_\beta + \tau_{b|d} \times (1 - \kappa_\beta)\right) \end{aligned} \tag{1}$$

Note that v_α, v_β, κ_α and κ_β are provided by the load model, and the $\tau_{x|y}$ are provided by the interference model.

[2] For $\tau_{y|x}$ to have the desired meaning, m_y should be a measurement of *link throughput*. However, link throughput and SNIR are approximately proportional — see [6].

[3] We assume here that whether or not interfaces a and b are transmitting are independent random events [7]. Then the probability that a is transmitting at any moment is v_a, and the probability that b is transmitting and a is listening at any moment is: $(1 - v_a) \times v_b$.

[4] That is, the interference caused by either interface c or interface d.

[5] The probability of two linked interfaces transmitting at the some time on an 802.11 mesh network can be as high as 7% — see [8], [9].

3 Adjusting the Channels

Each node is seen as an agent. The multiagent system exploits the distinction between proactive and reactive reasoning [10]. Proactive reasoning is concerned with planning to reach some goal. Reactive reasoning is concerned with dealing with unexpected changes in the agent's environment. The reactive logic provides an "immediate fix" to serious problems. The proactive logic, that involves deliberation and co-operation of nearby nodes, is a much slower process.

An agent (i.e. a node, or physically a router) with omnidirectional interfaces has three parameters to set for each interface: [1] The channel that is assigned to that interface; [2] The interfaces that that interface is linked to, and [3] The power level of the interface's transmission. Methods are describe for these parameters in the following sections. The following section describes how these three methods used combined in the proactive logic algorithm. The following methods all assume that there is a load balancing algorithm.

Informally the proactive logic uses the following procedure:

- *Elect* a node a that will manage the process
- *Choose* a link α from a to another node — precisely a trigger criterion (see below) permits node a to attempt to improve the performance of one of its links $\alpha \ni a$ with a certain priority level.
- *Measure* the interference
- *Change* the channel setting if this leads to a mean improvement for all the agents in a's interference range

The following is a development of the ideas in [2].

```
**choose** node $a$ **at** time $t-2$;
**set** $V_a = \cup_{n \in S_a} S_n$;
$\forall x \in V_a$ **transmit** "**propose organise**$[a,x,p]$";
**unless** $\exists x \in V_a$ **receive** "**overrule organise**$[a,x,q]$" in
                $[t-2,t-1]$ where $q > p$ **do** {
  $\forall x \in V_a$ **transmit** "**propose lock**$[a,x,t,t+1]$";
  **if** $\forall x \in V_a$ **receive** "**accept lock**$[a,x,t,t+1]$" in $[t-1,t]$
  **then** {
     **unless** $\exists x \in V_a$ **receive** "**reject lock**$[a,x,t,t+1]$"
          **do** {improve $a$;}
  }
}
where: improve $a$ = {
   **choose** link $\alpha \ni a$ on channel $\Gamma_\alpha^t$;
   **set** $B \leftarrow \sum_{\beta \in S_\alpha} f(\alpha \mid \beta) + \sum_{\beta \in S_\alpha} f(\beta \mid \alpha)$;
   **if** (feasible) **re-route** $\alpha$'s traffic;
   **for** $\Gamma_\alpha = 1, \ldots, K, \Gamma_\alpha \neq \Gamma_\alpha^t$ **do**{
      **if** $\sum_{\beta \in S_\alpha} f(\alpha \mid \beta) + \sum_{\beta \in S_\alpha} f(\beta \mid \alpha) < B \times \varepsilon$ **then**{
         $\Gamma_\alpha^{t+1} \leftarrow \Gamma_\alpha$;
         **selflock** node $a$ **in** $[t+1,t+k]$;
```

```
            **break**;
        };
    };
    $\forall x \in V_a$ **transmit** "α's interference test signals";
    **apply** load balancing algorithm to $S_a$;
}
```

The statement **selflock** is to prevent a from having to activate the method too frequently. The constant $\varepsilon < 1$ requires that the improvement be 'significant' both for node a and for the set of nodes S_a. The stability of this procedure follows from the fact that it produces a net improvement of the interference cost within S_a. If a change of channel is effected then there will be no resulting change in interference outside S_a.

The above method reduces the net observed inference cost in the region V_a. It does so using values for the variables that appear on the right-hand side of Equation 1. If those values are fixed then the method will converge. The method above suggests the possibility that traffic is re-routed during the reassignment calculation — this is not essential.

3.1 Results and Discussion

The interference cost reduction for a link discussed herein is measured as the difference between absolute interference (AI) values obtained before the channel assignment process and after the channel assignment process. For example, if $AI_{before} = 5$ and $AI_{after} = 4$ the absolute difference is $AD = 1$ which is 20% decrease in the absolute interference. Consequently, the performance is always expressed as a percentage of the decrease. Our simulation studies consider realistic scenarios of different node densities and topologies in a typical wireless mesh network hence are more reflective of evaluating the true performance of the algorithm. In these studies the mean of interference cost (IC) reduction across all topologies and network (node) densities obtained is 36.7.

Impact of typical topologies on the interference cost. Figure 2(a) shows the variation in the interference cost reduction as a function of network topology across different node densities. It can be deduced that the impact of the topologies on the performance of the algorithm (i.e. in terms of interference cost reduction) is insignificant. The mean of IC reduction calculated from the data obtained shows that the topology with the smallest average IC reduction is the completely random with a mean of 36.02 and topology with the most IC reduction is the random grid with a mean of 37.12. The difference in performance between best and worst case is just 1.1 which confirms that the performance of the algorithm is almost completely independent of the type of topology.

Performance Comparison across the Network. In this study, we obtained interference cost (IC) in different regions of the MR-WMN for the same set of links before and after the self-organisation algorithm is invoked. Comparison of the results obtained is shown in Figure 2(b) where the Interference cost is on the X-axis. From Figure 2(b) we can see that there were no nodes (red dots) that caused more interference after the self-organisation than it had caused before (blue dots) the self-organisation was invoked.

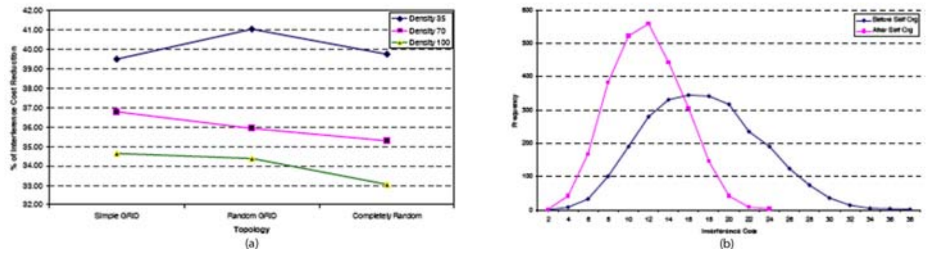


Fig. 2. (a) Interference cost reduction as a function of topologies. (b) Comparison of *IC* across the network before (blue) and after (red) self-organisation.

4 Adjusting the Links

The the first path-reduction algorithm is precisely the same as the algorithm in Section 3 but with the following 'improve' methods.

Link adjustment with known traffic load. Suppose that node a has interference range S_a. Let M_a be the set of nodes in S_a excluding node a. Then use the method in Section 3 with the following 'improve' method:

```
improve $a = \{$
  for link $\alpha \ni a$, where $\alpha = [a,b]$
  suppose $\alpha$ is on channel $\Gamma^t_\alpha$;
  set $B \leftarrow \sum_{\beta \in S_\alpha} f(\alpha \mid \beta) + \sum_{\beta \in S_\alpha} f(\beta \mid \alpha)$;
  if (feasible) re-route $\alpha$'s traffic;
  set $\gamma \leftarrow \alpha$;
  for $y \in M_a$ do $\{$
    for $\Gamma_{[a,y]} = 1, \ldots, K$, do $\{$
      if $\sum_{\beta \in S_a} f([a,y] \mid \beta) + \sum_{\beta \in S_a} f(\beta \mid [a,y]) < B \times \varepsilon$
       then $\{$
          set $\gamma \leftarrow [a,y]$;
          selflock node $a$ in $[t+1, t+k]$;
          break;
      $\}$;
    $\}$;
  $\}$;
  $\forall x \in V_a$ transmit "$\gamma$'s interference test signals";
  apply load balancing algorithm to $S_a$;
$\}$
```

Trigger for attempting to adjust a link with known traffic load. Consider a mesh with known traffic load. Suppose that the load balancing algorithm has allocated load to links on the mesh, and let link $(a,b) = \arg\max_{x \in N^t_a} \rho(x)$. If replacing (a,b) with (a,x) would mean that there exists a cut through the mesh that traverses (a,x) and that all other links on that cut have a load $< \rho(a,b)$ then let node a initiate the link adjusting procedure. Likewise if replacing (a,b) with (y,b).

4.1 Reactive Logic

The relationship between the reactive and proactive logics is determined by:

```
**if event** [link $\alpha$ is broken] **then** {
    **activate** [activate the *Reactive Method* for link $\alpha$];
    $\forall x \in \alpha$ **if state** [node $x$ locked by "**accept lock**[$a,x,s,t$]"
    **then** {**transmit** "**reject lock**[$a,x,s,t$]";}
}
```

where the *Reactive Method* is as follows; it simply fixes disasters as they occur possibly with a configuration that is less satisfactory than the prior. It has no implications for neighbouring interfaces, and so it presents no instability issues.

Link adjustment with unknown traffic load. Suppose that node a has interference range S_a. Let M_a be the set of nodes in S_a excluding node a. For nodes $x,y \in S_a$, let $c(x,y)$ denote the cost[6] of the least cost path that connects x and y. We assume that: $(\forall x,y)c(x,y) = c(y,x)$, and that if the least cost path between nodes u and v is a subset of the least cost path between x and y then $c(u,v) \leq c(x,y)$. Let N_a^t be the set of links in S_a at time t, and $N_a^t(\ominus[a,x],\oplus[a,y])$ denotes the network configuration with link $[a,x]$ replaced by $[a,y]$. Let $C(N_a^t)$ denote the cost of the path of greatest cost in S_a: $C(N_a^t) \triangleq \max_{x,y \in S_a} c(x,y)$. Choose the pair of nodes b and c by:

$$(b,c) = \arg \min_{(x,y) \mid [a,x] \in N_a^t, y \in M_a} C(N_a^t(\ominus[a,x],\oplus[a,y]))$$

and swap link $[a,b]$ for link $[a,c]$ if:

$$C(N_a^t(\ominus[a,b],\oplus[a,c])) < C(N_a^t) \times \varepsilon$$

where $\varepsilon < 1$ is a threshold constant [11].

4.2 Second Path Reduction Algorithm

In our second path reduction algorithm only link substitution method is used which makes it different from our previous PR algorithm in which link substation is used along with link addition. Another distinction is that previously we selected only those substituted links that would not increase the IC whereas in the current algorithm we select links irrespective of their effect on the IC.

The outline of our algorithm is succinctly given here: An initiator node selects one of its available radio interfaces on the basis of strongest transmission power, the selected interface creates a list of available interfaces in its communication range (locality principle), from these interfaces those that have a path length to the portal node longer than the shortest path are filtered out. The remaining ones are short listed and an interface

[6] The precise meaning of this cost function does not matter. It could be simply the number of hops, or some more complex measure involving load and/or interference.

```
For node $n_x \in N$ select one of its interfaces $i_x$
that is free and has the strongest signal of all
its free interfaces;
if $i_x = \varnothing$
        for $n_x$ set $blockFree \leftarrow bc;$
        $end;$
else
        set $blockBusy \leftarrow bc$;
set $I_{free} \leftarrow communicationRange(i_x, I)$ ;
if $I_{free} = \varnothing$
        $endAction.$
set $i_y \leftarrow bestSNR(I_{free})$
if $i_y = \varnothing$
        $endAction.$
$l_{i_x i_y} \leftarrow createLink(i_x, i_y)$
for $n_x$ $linkCounter\ \ linkSubCounter + 1;$
$remove\ (l_{i_u i_v});$
$reset\ (i_u, i_v);$
$set\ p \leftarrow newShortestPath(n_x)$
end.

$endAction$
        for $n_x$
                set $blockBusy \leftarrow 0;$
                set $blockFree \leftarrow bc;$
$end.$
```

N	a set of all nodes
I	a set of all interfaces
L	a set of all links
I_{free}	a subset of interfaces
i_x	an available interface.
n_x	a node which contains i_x.
i_y	an available interface.
n_y	a node which contains i_y.
i_u	an interface with shortest path on n_x
n_v	a node that contains i_v.
i_u	an interface on n_z that is linked to i_u
p	a shortest path for a n_x
$l_{i_u u_v}$	a link between i_u *and* i_v
bc	a blocking constant

Fig. 3. Second path reduction algorithm

from these that offers the best SNIR is selected; the new link between the two interfaces is created and the previous shortest path link is switched off and its interfaces operational attributes are reset to their default values. This process occurs simultaneously across the multiagent system. A more formal description of the algorithm is given in Figure 3. The functions in the algorithm are:

- cummuncationRange(interface, set of interfaces): This function selects all interfaces from the set of interfaces that are free and within the range of interface.
- shortestPath(p, interface, set of interfaces): This function selects a set of interfaces that has a shortest path from the interface to the set of interfaces. If the path between the interface with a shortest path and the interface is not shorter than path p the function returns ∅. Otherwise it returns a set of interfaces with a shortest path.
- bestSNIR(set of interfaces): This function selects an interface with a best SNIR from the set of interfaces. If there is more than one such interface this function randomly selects one.
- createLink(interface A, interface B): This function creates a link between agent A and agent B and returns a newly created link.
- remove(link): This function removes a link from the set L.
- reset(interface A, interface B): This function resets attributes of agent A and agent B to default attributes.
- newShortestPath(node): This function returns a shortest path value for the node.
- A node is either *locked* or *unlocked*. A locked node is either locked because it has committed to lock itself for a period of time on request from another node,

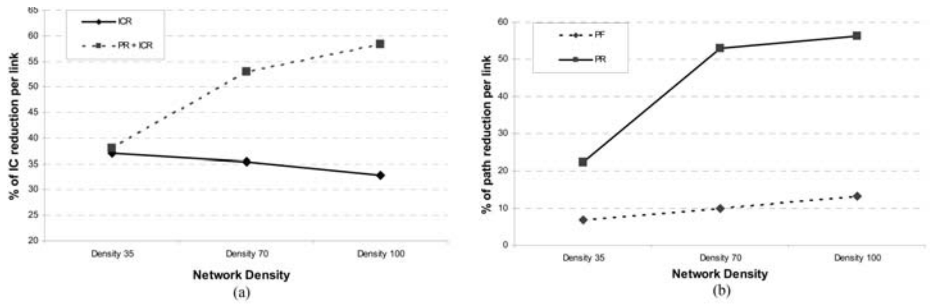


Fig. 4. (a) Result (comparative study) showing the increase in IC reduction in comparison to ICR only algorithm. (b) First versus the second algorithm for path reduction through link substitution.

or it is self-locked because it has recently instigated one of the self organisation procedures. A locked node is only locked for a "very short" period. This is simply to ensure that no more than one alteration is made during any one period which is necessary to ensure the stability of the procedures.

4.3 Results and Discussion

We present below some of the key results that we have obtained to illustrate the performance of the second path reduction algorithm. Figure 4(a) shows a graph of ICR vs. network density when just the algorithm for ICR has been invoked and both the path reduction and ICR algorithms has been invoked. This comparative study clearly shows that without the invocation of the path reduction algorithm the effect of ICR process results in much higher IC. A reason for this is twofold. Firstly, because of an increased number of interfaces that the initiator node can use and secondly, because of the additional mechanism that selects the interface based on the best SNIR value. Furthermore, as the network density increases the performance of the path reduction followed by ICR significantly increases whereas just the performance of the ICR algorithm on its own slightly decreases. Figure 4(b) compares the path length reduction that is achieved by using the first algorithm and the second path length reduction algorithm. The second algorithm is much more effective than the first by a factor of 2 to 5 times.

5 Conclusion and Future Work

We have described a multiagent system based self-organising algorithm for multi-radio wireless mesh networks (MR-WMN) that can operate on any radio technology. The multiagent system ensures scalability by progressively assigning the channels to nodes in clusters during the WMN system start up phase. The stability is offered by means of the proactive and reactive logic of the system. These attributes were validated through analysis and simulation. Through the work described in this report we have examined motivation and developed a multiagent system for the topological control of MR-WMN. The goal of this algorithm is to increase the number of shortest paths to the portal nodes without adversely effecting interference cost. In addition to interference cost reduction implementation of this algorithm on MR-WMN further improve the system capacity.

Our future work will be focused on the development of our Java framework that is multi threaded so each node is represented as an independent thread. We believe that this will enable us to develop algorithms for tuning the capacity of the network links according to fluctuations in demand by mobile users.

References

1. Raniwala, A., Chiueh, T.c.: Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network. In: Proceedings IEEE Infocom 2005. IEEE Computer Society, Los Alamitos (2005)
2. Ko, B.J., Misra, V., Padhye, J., Rubenstein, D.: Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks. Technical report, Columbia University (2006)
3. Mishra, A., Rozner, E., Banerjee, S., Arbaugh, W.: Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage. In: ACM/USENIX Internet Measurement Conference (2005)
4. Mishra, A., Shrivastava, V., Banerjee, S.: Partially Overlapped Channels Not Considered Harmful. In: SIGMetrics/Performance (2006)
5. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh netowrks: a survey. Computer Networks, 445–487 (2005)
6. Vasudevan, S.: A Simulator for analyzing the throughput of IEEE 802.11b Wireless LAN Systems. Master's thesis, Virginia Polytechnic Institute and State University (2005)
7. Leith, D., Clifford, P.: A self-managed distributed channel selection algorithm for wlans. In: Proceedings of RAWNET, Boston, MA, USA, pp. 1–9 (2006)
8. Duffy, K., Malone, D., Leith, D.: Modeling the 802.11 Distributed Coordination Function in Non-saturated Conditions. IEEE Communication Letters 9, 715–717 (2005)
9. Tourrilhes, J.: Robust Broadcast: Improving the reliability of broadcast transmissions on CSMA/CA. In: Proceedings of PIMRC 1998, pp. 1111–1115 (1998)
10. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: The foundation for the organization of multiagent systems. In: Mařík, V., McFarlane, D.C., Valckenaers, P. (eds.) HoloMAS 2003. LNCS, vol. 2744, pp. 71–80. Springer, Heidelberg (2003)
11. Ramachandran, K., Belding, E., Almeroth, K., Buddhikot, M.: Interference-aware channel assignment in multi-radio wireless mesh networks. In: Proceedings of Infocom, Barcelona, Spain, pp. 1–12 (2006)

Mobility Model for Tactical Networks

Milan Rollo and Antonín Komenda

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
Technická 2, Prague 6, 166 27 Czech Republic
{rollo,komenda}@labe.felk.cvut.cz

Abstract. In this paper a synthetic mobility model which represents behavior and movement pattern of heterogeneous units in disaster relief and battlefield scenarios is proposed. These operations usually take place in environment without preexisting communication infrastructure and units thus have to be connected by wireless communication network. Units cooperate to fulfill common tasks and communication network has to serve high amount of communication requests, especially data, voice and video stream transmissions. To verify features of topology control, routing and interaction protocols software simulations are usually used, because of their scalability, repeatability and speed. Behavior of all these protocols relies on the mobility model of the network nodes, which has to resemble real-life movement pattern. Proposed mobility model is goal-driven and provides support for various types of units, group mobility and realistic environment model with obstacles. Basic characteristics of the mobility model like node spatial distribution and average node degree were analyzed.

Keywords: Mobility model, simulation, tactical networks.

1 Introduction

In the future field operations like a disaster relief operations, battlefield operations, or reconnaissance/surveillance will be carried out by mixed human-machine teams. Those teams may consist of a high number of autonomous robots, unmanned aerial vehicles, unattended ground sensors and humans. Operation of these heterogeneous teams brings challenging research problems: *(i)* missions usually take place in environments without pre-existing communication infrastructure and units are thus connected by ad hoc communication networks, *(ii)* there is no central component where all the information about field operation units, like their positions, abilities and plans can be stored and accessed by other team members and *(iii)* to operate efficiently, units need to interact one with the other to coordinate their activities, improve their situational awareness or to be able to plan future tactical missions.

Communication network thus has to deliver large volume of data (video and voice streams, application data). Communication resources must be utilized in effective way to ensure quality of service provided to the users (like message delay and packet drop ratio) while minimizing the energy consumption and interference. To ensure the efficient utilization of resources various algorithms and interaction protocols are used, namely the topology control protocols, message routing protocols, negotiation protocols, etc. It is necessary to test these protocols thoroughly during the development phase to verify their robustness and scalability. These test are usually carried out using software simulations, because of their scalability, repeatability and ability to test scenarios under various settings and parameters.

Communication protocols built on the top of physical communication infrastructure have several limitations. Links in a wireless communication network have a limited range and bandwidth, they are subject to signal interference and the transmission over the link affects bandwidth of the links in its surrounding. Parameters of the communication links depend mainly on positions of network nodes. During the development and test it is thus necessary to simulate node mobility.

Existing mobility models are not suitable for simulation of complex dynamic movement patterns which occur in tactical networks consisting on numerous types of units operating in real environment with obstacles. Goal of this paper is to propose a goal-driven mobility model which will resemble mobility patterns of heterogeneous units during real-life field operations.

The rest of the paper is organized as follows. In Section 2 existing mobility models are described. In Section 3 problem formulation is given and our novel approach to solving the problem is described. Experimental evaluation of features of the proposed mobility model is described in Section 4. We conclude with a summary and an outline of future work is Section 5.

2 State of the Art

Mobility of the units has to be taken into account during the network/mission design process. Node mobility models are used to simulate the real world behavior to determine if the proposed protocols will satisfy given criteria when implemented. As the real-life movement patterns (traces) are very difficult to obtain, synthetic mobility models are used during the simulation and for verification of networking protocol features. Mobility models for ad hoc/sensor networks should try to fulfill two goals which are often conflicting [1]:

- resemble real-life movements – ad hoc and sensor networks are used in wide range of domains with various movement patterns, e.g. disaster relief operations, vehicular motion, sensors carried around by the ocean flows, movement of group of tourists, etc. Each of these domains usually requires its specific mobility model.
- be general/simple enough for simulation and formal analysis – to keep simulation times reasonable, mobility models should be simple enough. Moreover, using relatively simple mobility modes allows formal analysis of their behavior with respect to fundamental network parameters and influence of mobility on performance of networking protocols.

In [2] authors propose classification of mobility models based according to different dependencies and restrictions:

– Random based - nodes move randomly without any restrictions or dependencies on environment and other nodes;
– Temporal dependencies - actual movement of the node is influenced by its previous movement;
– Spatial dependencies - movement of the node is affected by other nodes in its surrounding (e.g. group mobility, potential fields, etc.);
– Geographic restrictions - nodes are restricted to move only in some predefined areas;
– Hybrid characteristics - mobility model combines some or all of previous characteristics.

Below the mobility models most widely used for simulation of ad hoc wireless networks are described. For more detailed description see e.g. [2] [3].

The most widely used mobility model for ad hoc networks is the Random Waypoint Model (RWP) which has been introduced in [4]. In this model each node moves independently of each other and the environment is assumed to be obstacle-free. In RWP model each node is assigned an initial location and chooses a destination point (the *waypoint*) uniformly at random in region R. Example can be e.g. movement of people at the airport terminal or town square. Variant of this mobility model is a Random Direction Model which (unlike the RWP) maintains a uniform node spatial distribution during the simulation.

Example of mobility model with geographic restrictions can be the *Manhattan mobility model* [5], which emulates movement of people in metropolitan traffic. Manhattan-like map consists of horizontal and vertical streets. Nodes can move along the streets in both directions and any time they enter an intersection they randomly choose a new direction of movement - they can continue in the same direction, or turn left/right. Another map-based mobility is a *freeway mobility* model [5], which represents movement of cars on freeways. Several freeways, each composed of varying number of lanes in both directions, are located within the region R. Each node is restricted to its lane on the freeway and its velocity is temporarily dependant on previous velocity.

In many scenarios nodes are expected to move in groups (e.g. the battlefield scenarios or groups of tourists in the city). Various group-based mobility models has been introduced to model these scenarios. In case of the *reference point group mobility* model [6] a subset of the network nodes is defined as a set of group leaders $N_L \subset N$. Rest of the nodes is assigned randomly to the group leaders. After the simulation start, the group leader starts to move according to one of the previous mobility patterns and other group members follow the leader. Another instance of group-based mobility, *diamond group mobility* model which is suitable for ad hoc network in military scenarios is described in [7].

In [8] Aschenbruck et. al propose a hybrid mobility model for disaster area scenarios. Authors identify following main requirements for the mobility model: (i) heterogeneous velocity, (ii) tactical areas, (iii) optimal paths, (iv) obstacles,

(v) units join and leave the scenario and (vi) group movement. Their mobility model defines several areas typical for the catastrophe situations like the incident site or casualties treatment area and specialized units which operate within these areas (e.g. transport units or firefighters).

3 Mobility Model for Tactical Networks

In this section we propose a synthetic mobility model which is suitable for tactical networks. This model is a generalization of the disaster area mobility model described by Aschenbruck et. al in [8] and is inspired by several other mobility models, especially the group-based mobility.

3.1 Problem Formulation

Santi in [1] formulates the problem of node placement as follows:

Let N be a set of nodes with $|N| = n$. Nodes operate in a bounded region R. Let assume that R is a d-dimensional cube of side l. Formally, $R = [0, l]^d$ for some $l > 0$, where $d = 1, 2, 3$. Location of any node $u \in N$ in R is denoted $L(u)$ and is expressed in d-dimensional coordinates. Function $L : N \rightarrow R$ maps every node of the network to its physical location. In case of mobile networks, the physical location of nodes changes in time. Mobility can be represented by adding additional parameter into the placement function L, the time instant t (assuming that nodes move within R). Function $L : N \times T \rightarrow R$ assigns to every node of N and to any time $t \in T$ a set of d-dimensional coordinates representing the physical node's location at time t. A d-dimensional mobile ad hoc network is then represented by pair $M_d = (N, L)$.

Synthetic mobility models generate the placement function $L : N \times T \rightarrow R$ for each node $n \in N$ according to some algorithm. Each algorithm relies on several internal parameters which can in some cases significantly affect the placement function. These parameters have to be set carefully with respect to the real-world scenario the mobility model has to simulate.

3.2 Types of Nodes

Tactical networks are complex systems which consist of heterogeneous mixture of goal-driven nodes. Behavior of resulting mobility model depends heavily on the type of nodes involved in the mission, as each type has its individual mobility pattern. Generally we can distinguish following types of nodes:

– Autonomous robots – all autonomous computer-controlled units, like robots, aerial assets or ground vehicles. Each robot is running an internal path planner which can plan movement trajectory from the current position to the destination point. This trajectory can be described by a set of waypoints, time constraints in these waypoints and path elements between them (straight elements, turns, etc.). Robots autonomously select and fulfill tasks accordingly to the overall mission goal.

- Wireless sensors – they remain stationary during the entire simulation time. They can be distributed either randomly over the whole region R or according to some pattern, simulating the case of being spread from a moving airplane or vehicle.
- Humans – like the autonomous robots humans participate on the task execution process. However their movement trajectory cannot be easily described by a set of waypoints, as their exact movement is highly unpredictable.
- On-demand communication nodes – these nodes represent special units dedicated to improve the topology of the communication network. They proactively move to the positions where they can improve connectivity of the network. Their movement is fully controlled by the networking protocols and they thus cannot be modeled by this low level algorithm.

3.3 Mobility Model Description

Tactical network mobility model can be formalized as follows:

In environment operate m different types of nodes (e.g. humans, robots, UAVs, etc.). Each type can be described by following parameters:
$type = \langle type_{id}, v[v_{min}, v_{max}], mobility\ pattern \rangle$, where v_{min} and v_{max} are minimum and maximum velocity of the node and mobility pattern describes allowed maneuvers with respect to the physical constraints of the node. E.g. ground units are allowed to stop and turn in place while UAVs must move continuously with speed higher then v_{min} and turn maneuvers must respect minimum turn radius of the UAV.

Let N be a set of all nodes with $|N| = n$. Subset $N_s \subset N$ represents set of all stationary sensors. Number of sensors $n_s = |N_s| \in [0, n]$. If $n_s = 0$ no wireless sensors are placed in R, in case of $n_s = n$ problem can be transformed to wireless sensor network (WSN). Set of stationary nodes can be selected randomly using the probability parameter p_{stat} (in the case of the random node placement) or can be predefined (when nodes were spread according to some pattern).

Remaining nodes belong to the set of mobile nodes $N_m = N \setminus N_s$. Each node is described by a following tuple $\langle type_{id}, R_n \rangle$, where $type_{id}$ represents type of the unit and $R_n \in R$ is a set of areas the node is allowed to operate in. At the beginning of the simulation nodes can be placed uniformly at random at the any of the area R_n. Number of members of individual types can either be given at the beginning of the simulation (e.g. number of robots is known) or can be set according to some parameter p_{type}, which represents the probability that a node belongs to given type.

Areas are defined as a set of polygons in case of 2D scenarios and as a set of polyhedrons in 3D. Environment may contain a set of obstacles O. Nodes have to avoid these obstacles during their movement while respecting the areas they are allowed to operate in. Specific path planning algorithm thus has to be implemented to compute the node trajectories for each mobility pattern. Typically, methods of robot motion planning can be used to plan trajectories of ground units (robots, UGVs, cars) and specific 3D planning algorithms (e.g. [9]) have to be used for air traffic simulation (airplanes, UAVs, helicopters).

Area in which unit can operate is than defined as a union of all allowed areas minus the obstacles $R_{allowed} = \bigcup_{R_n} - \bigcup_{O}$. Example of such an area is shown in Figure 1, where $R_n = \{R_1, R_2, R_3\}$ and obstacles are represented by polygons $O = \{O_1, O_2, O_3\}$.

As stated above, nodes try to fulfill a set of tasks. In real-life scenarios several units usually have to participate on the task execution process. Task tk thus can be described by a tuple $tk = \langle L(tk), P(type_{id_1}, \ldots, type_{id_m}), t_{tk}, p\rangle$, where function L gives the physical location in R where the task must be executed, set of participants P specifies a number of nodes of each type (with m different types of nodes) required to fulfill the task. Parameter t_{tk} represents the duration of the task and p is a task priority. Maximum number of each type of participating nodes parameters tk_{type_max}. Batch of random tasks TK is generated. Number of tasks in one batch is given by the parameter tk_{batch} – $|TK| = tk_{batch}$.

Function

$$available(u) = \begin{cases} 0 & \text{if node } u \in N_m \text{ is already assigned to some task } tk \\ 1 & \text{otherwise} \end{cases}$$

With given sets of nodes and tasks the mobility model works as follows:

1. Select random task $tk \in TK$ and try to assign required number of nodes to the task. Path planner filters out nodes that cannot reach the task location. Available nodes are ordered by their distance from the task location $L(tk)$ and the closest ones are assigned to the task. If sufficient number of nodes is not available, other task from the TK is selected. In case that none of the tasks can be executed, wait until some other task is completed and more nodes will become available.
2. When new task is assigned to the group of nodes, it is removed from the batch of tasks $TK = TK \setminus tk$ and all the group members start to move to the *rendezvous point*. This node is placed in the middle of the group of nodes assigned to the task (average value of their coordinates). Note that the predefined rendezvous point can be used as well. Nodes do not move exactly to that point, but stay somewhere within the ϵ-neighborhood to simulate some pattern. In more realistic scenarios form of that pattern can be specified in details.
3. Individual nodes move according to their node mobility. Following mobility models when approaching the rendezvous point were used in our simulation:
   - robotic nodes move along a straight line with a velocity v_{max};
   - humans can either move or remain stationary in given time step, which simulates their engagement in other event, e.g. analysis of the map updates received. When their move, their motion is a variant of the Brownian-like movement – their position in the next time step is in square of side $2m$, which is not centered at the current node location at this case, but current location is in the center of square side and square lies on the joint of current position and rendezvous point, see Figure 2. Parameter m represents the average motion speed of the human.

Different mobility models can be used for each node type. Example of unit movement is shown in Figure 3.

4. After the group is formed, group leader node is selected and all members have to agree on a common group velocity. Then the group starts to move to the task location point $L(tk)$. Group velocity is set to the minimum from the v_{max} of the slowest node and m. All the humans try to stay within some δ-vicinity of the group leader. If they are getting out of this region they temporarily modify their m parameter.
5. When all the group members reach the ϵ-neighborhood of $L(tk)$, they remain stationary for the time interval t_{tk} to simulate task duration. After that time period their availability status is set to 1 and new task assignment process is invoked.
6. If $TK = \{\emptyset\}$, generate new batch of tasks. Two variants of batch generation process were studied – (i) *continuous*, when new batch is generated immediately after the last task from previous batch is assigned to some unit(s) and (ii) *delayed*, when next batch is generated once the last task from the previous batch is completed. By generating tasks in batches sequential nature and precedences of tasks are simulated.

3.4 Model Extensions

Mobility model can be extended with a feature which allows units join and leave the scenario as mentioned in [8]. This feature is very important for testing of higher-level control algorithms, as it brings additional dynamics to the system. It could be used to test to test ability of routing protocols to dynamically modify the routing tables, adaptivity of clustering algorithms or response time of topology control protocols.

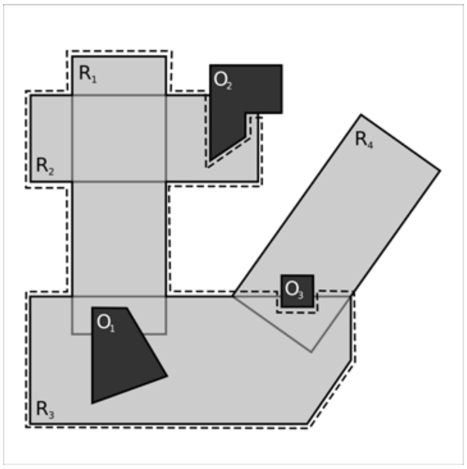


Fig. 1. Example of the field operation scenario. Units are allowed to operate only within the predefined areas and have to avoid the obstacles.

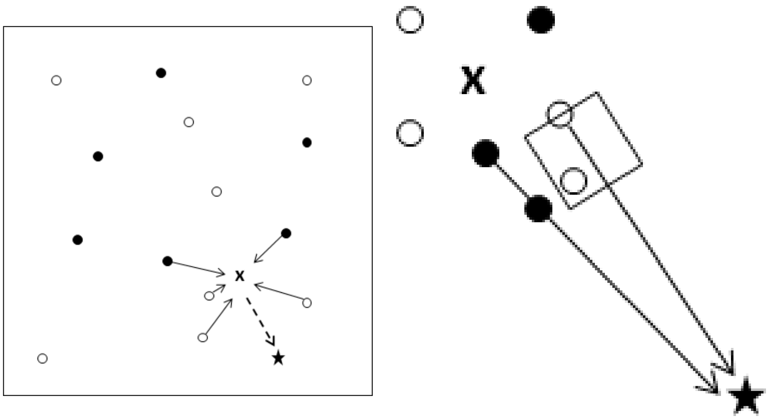


Fig. 2. Example of topology network mobility model: 3 humans and 2 robots that will participate on the task are moving towards the rendezvous point (a), detail of the motion model (b)

Typically we distinguish two basic cases when units leave the scenario:

1. randomly - units are disposed at random time to simulate their failures, to simplify the simulation process remaining nodes continue in task execution and do not try to look for additional nodes to substitute the missing one;
2. when task is completed - units leave the predefined areas (i.e. task location point $L(tk)$ is not selected randomly, but rather set to some predefined point on a border of allowed area.) or their communication devices are turned off.

Units may join the scenario in a similar way - either they enter the area at predefined points on area border or turn on their communication devices (which allows to randomly deploy nodes inside the area).

According to the simulation scenario setup number of active nodes may not excess predefined number of nodes n. In such a case several nodes have to leave the scenario before new nodes are allowed to join.

4 Experimental Evaluation

Set of experiments was carried out to illustrate basic characteristics of the mobility model. Nodes were operating in 2D flat, obstacle-free environment, area was a square 1000x1000m. Communication range was set to 100m.

As could be seen in Figure 4, critical transmitting range (CTR) for connectivity for proposed mobility model is higher for the same number of nodes deployed over the area compared to the RWP mobility model. Obviously this is caused by the fact that nodes form groups and move in close formations in contrast to the independent random node movement in case of RWP. Generally there are dense parts of the area that represent the clusters and whole network can be seen as a sparse configuration of clusters. Longer distances between the clusters lead to higher required CTR to prevent the network partitioning. In experiments for the tactical network mobility model one half of the nodes were robots and the second half humans, maximum group size was $\frac{n}{10}$ and each batch consisted of $|TK| = 10$ tasks.

Average node degree distribution (Figure 4 (b)) shows that average number of neighbors increases with the number of nodes required to complete the task. In case of RWP the distribution has a small variance, while with increasing number of nodes in group the variance grows.

Average length of the trajectories is shorter than in the case of RWP mobility (see Figure 3), because for each task a group of nearest units is chosen. This matches the real-world behavior, where closest units are chosen as well to minimize the travel time and energy consumption.

Spatial density distribution varies for each batch generation process:

– Continuous batch generation – in this case a node density is similar to one of RWP model. During the detailed analysis of the RWP model it has been shown, that model generates node spatial distribution which is independent of the initial node placement and that the nodes are concentrated in the center of the region R (*border effect*) [10].Strength of the border effect in this case depends on the number of nodes participating on task execution. The higher is the ratio of participating nodes compared to the total number of nodes the more distinct is the effect, because only few groups can be formed and they have to travel around whole area. On the other hand, if only a few nodes are required to fulfil the task, robots travel short distances and node density is more uniform.
– Delayed batch generation – as in the previous case, characteristic of the node spatial density strongly depends on the size of the groups required to complete the tasks. Furthermore it also depends on the number of tasks in a batch. If the number of tasks is low and small number of robots is required, we can observe *reverse border effect*, when nodes tend to stay around the border of the area. Reason for this behavior is, that nodes wait until completion of all tasks in the batch, after that new batch of tasks is generated and assigned to the nodes. But tasks are assigned to nodes in their vicinity and nodes close to the border are very likely to remain in their positions. With increasing number of tasks and size of the groups distribution becomes more uniform and for big groups the distribution is similar to the RWP model. Node spatial density for delayed batch generation process is shown in Figure 5.

So by adjusting model parameters and batch generation process various characteristics of node spatial densities can be achieved.

As in some other mobility models the probability distribution of the initial locations and velocities of the nodes differs from the distributions later in the simulation. There are in general two basic approaches how to deal with this problem – *(i)* discard an initial sequence of observations from simulation and *(ii)* use *steady-state* mobility generator which will choose initial locations and speeds from the stationary distribution [11]. In our case we have chosen the first

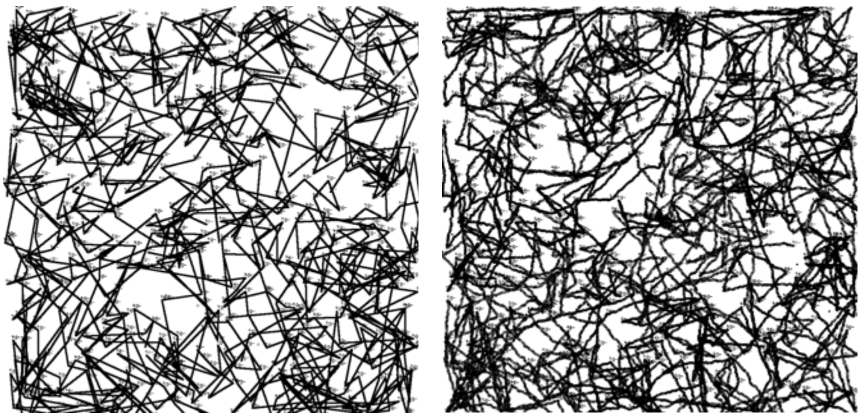


Fig. 3. Example of movement of robotic (a) and human (b) units

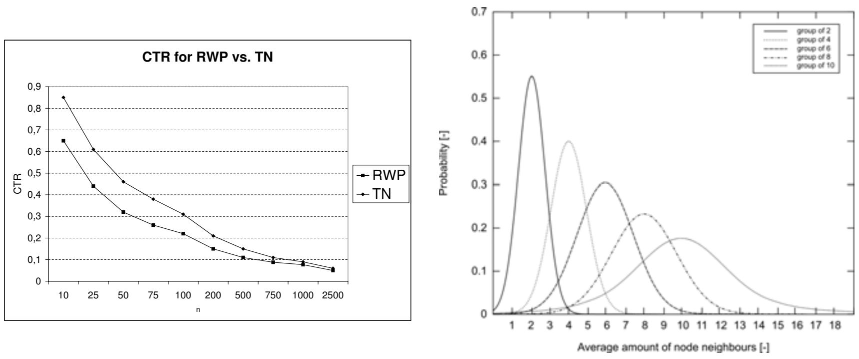


Fig. 4. CTR for connectivity - RWP mobility model vs. tactical network mobility model (a), average node degree distribution (b)

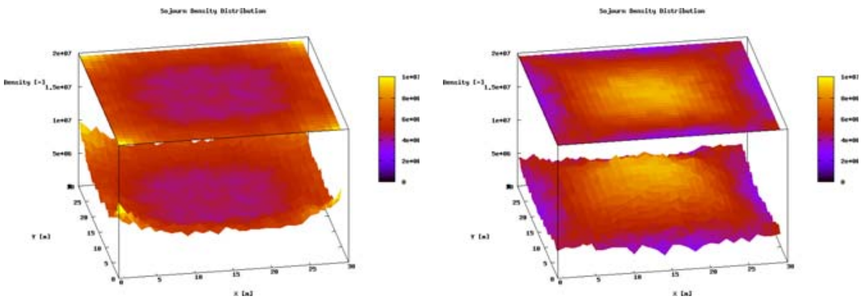


Fig. 5. Node spatial density distribution for delayed batch generation process. Figure shows situations for small groups and limited batch size (a) and for big groups (b).

approach and we discard a whole sequence at the beginning of simulation, until the first task is completed.

5 Conclusion

We have described a problem of modeling mobility of nodes in field operations and proposed a novel synthetic mobility model which addresses this problem. Unlike the other existing models it is able to simulate complex scenarios and is general enough to allow modeling of various scenarios. Behavior of nodes in our mobility model is goal-driven, i.e. fulfilment of common application scenario-related goals is simulated. Model is providing support for multiple types of mobile units as well as stationary wireless sensors. Depending on the generated goals nodes may form temporary groups and move in formations throughout the environment, which resembles real-life behavior patterns of units in tactical missions. Various features of the proposed model were studied and discussed, especially the role of goal-generating process on the overall behavior of the placement algorithm.

Acknowledgement

We gratefully acknowledge the support of the presented research by US Army project no. W911NF-08-1-0505 and research programme no. 1M0567 funded by the Ministry of Education of the Czech Republic.

References

1. Santi, P.: Topology control in wireless ad hoc and sensor networks. John Wiley & Sons Inc., Chichester (2005)
2. Aschenbruck, N., Gerhards-Padilla, E., Martini, P.: A survey on mobility models for performance analysis in tactical mobile networks. Journal of Telecommunications and Information Technology 2, 54–61 (2008)
3. Barrett, C.L., Drozda, M., Marathe, M.V., Ravi, S.S., Smith, J.P.: A mobility and traffic generation framework for modeling and simulating ad hoc communication networks. Scientific Programming 12(1), 1–23 (2004)
4. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In: Mobile Computing, pp. 153–181. Kluwer Academic Publishers, Dordrecht (1996)
5. Bai, F., Sadagopan, N., Helmy, A.: Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In: IEEE INFOCOM – The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, CA, pp. 825–835 (2003)
6. Wang, K., Li, B.: Group mobility and partition prediction in wireless ad-hoc networks. In: Proc. of IEEE International Conference on Communications (ICC 2002), vol. 2 (2002)
7. Ning, L., Yan, G., Chao, D., Jinlong, W.: Diamond group mobility model for ad hoc network in military. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2008. LNCS, vol. 5007, pp. 2754–2756. Springer, Heidelberg (2008)

8. Aschenbruck, N., Gerhards-Padilla, E., Gerharz, M., Frank, M., Martini, P.: Modelling mobility in disaster area scenarios. In: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems, pp. 4–12. ACM, New York (2007)
9. Šišlák, D., Volf, P., Pěchouček, M.: Accelerated a* path planning. In: Decker, Sichman, Sierra, Castelfranchi (eds.) Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 2009, pp. 1133–1134 (2009)
10. Bettstetter, C., Krause, O.: On border effects in modeling and simulation of wireless ad hoc networks. In: In Proc. IEEE Intern. Conf. on Mobile and Wireless Communication Networks (MWCN), pp. 20–27. Kluwer Academic Publishers, Dordrecht (2001)
11. Navidi, W., Camp, T.: Stationary distributions for the random waypoint mobility model. IEEE Transactions on Mobile Computing 3, 99–108 (2004)

Holonic Modelling of Large Scale Geographic Environments

Mehdi Mekni and Bernard Moulin

Cognitive Science Laboratory
Laval University, Department of Computer Science and Software Engineering
mmekni@gmail.com, bernard.moulin@ift.ulaval.ca

Abstract. In this paper, we propose a novel approach to model *Virtual Geographic Environments* (*VGE*) which uses the *holonic approach* as a computational geographic methodology and *holarchy* as organizational principle. Our approach allows to automatically build *VGE* using data provided by Geographic Information Systems (*GIS*) and enables an explicit representation of the geographic environment for Situated Multi-Agent Systems (SMAS) in which agents are situated and with which they interact. In order to take into account geometric, topologic, and semantic characteristics of the geographic environment, we propose the use of the holonic approach to build the environment holarchy. We illustrate our holonic model using two different environments: an urban environment and a natural environment.

1 Introduction

The importance of the environment as a *first-order abstraction* is particularly apparent for *Situated Multi-Agent Systems* (SMAS) [1]. SMAS are characterized by an explicit spatial environment called *Virtual Geographic Environment* (VGE) in which agents are situated. Indeed, such a representation must take into account the geometrical information corresponding to various geographic features. Moreover, this representation must qualify space by associating semantics with geographic features in order to allow spatial reasoning. *Geographic Information Systems* (GIS) provide such features but in an inefficient way when considering fast and automated spatial reasoning algorithms. In addition, modelling large scale geographic environments is a complex process [2]. Current approaches aim at considering the environment as a monolithic structure which considerably reduces the capacity to handle large scale, real world environments [3]. In order to build realistic SMAS we need an explicit environment representation which efficiently organises the geographic features, accurately captures the real world complexity, and reliably models large scale geographic environments.

The *holonic* approach has been successfully applied to a wide range of applications including industrial [4], military [5], environment monitoring [6], to name a few. Moreover, the *holonic* approach is a suitable organizational paradigm that may help for the space decomposition and organization of geographic features [3]. The term "*holon*" was originally coined by Arthur Koestler [7], basing it on the Greek word "*holos*" for "*whole*" and the suffix "*-on*" that denotes "*part*". According to Koestler, a holon is a fractal structure that is stable, coherent, and consisting of several holons acting as sub-structures. In this definition, a holon itself can be considered as part of a higher level architecture of holons. As an illustration in geographic context, a holonic decomposition of the geographic environment would consist of regions which in turn consist of groups of cells that can be further decomposed, and so on. Furthermore, the geographic environment may be part of a province and a country. None of these components can be understood completely without its sub-components or without the super component they are part of. Several holons, having each its own identity, can exist together as components of a given system. In this system, called a *holarchy*, holons are independent with respect to their subordinate parts and simultaneously dependent of parts of higher hierarchic levels. Thus, a holarchy denotes a hierarchical organization of holons having a recursive structure [7].

In this paper, we propose a novel approach to model *Virtual Geographic Environments* (*VGE*) which uses the *holonic approach* as a computational geographic methodology and the *holarchy* as an organizational principle. Our approach provides an exact representation of the geographic environment using GIS data. This representation is organised as a topological graph, enhanced with data integrating both quantitative data (like the geometry) and qualitative information (like the types of zones such as roads and buildings). This study focuses on the environment description and takes advantage of the holonic approach in order to address the following issues: 1) the way to efficiently and accurately model complex geographic spaces in order to build accurate VGE, 2) the way to structure, organize, and inform such a VGE in order to provide an easy and fast access to data describing the real world to situated agents to reason about it.

Next section presents a discussion of related works on modelling and representation of geographic environments. In Section 3, we introduce the geographic concepts used in our model. Section 4 presents our novel holonic modelling approach of large scale geographic environments. In order to demonstrate the generic aspect of the proposed model, we show in Section 5 two different experimental models. The first refers to a geographic holonic model representing the *Montmorency* experimental forest (*St.Lawrence Region, Quebec, Canada*). The second illustrates the urban holonic model of a part of Quebec city (*QC, Canada*). Section 6 concludes the paper and presents future work.

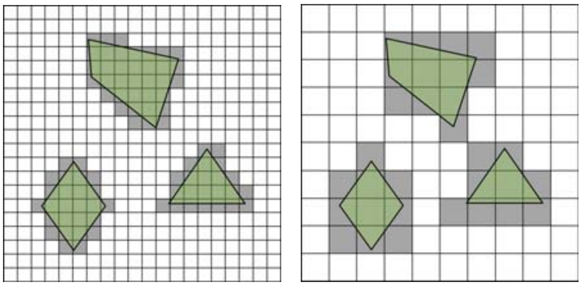
2 Related Work

While most works on *Multi-Agent Systems* (MAS) focus on the design of autonomous agents organisations, few research works have been found addressing the environment description and representation. Farenc proposed in [8] an informed environment dedicated to urban life simulation, which is based on a hierarchical decomposition of a urban scene into environment entities providing geometrical information associated with semantic notions. This approach puts forward the importance of integrating additional information in order to enrich the environment description but lacks a systematic method to decompose the geographic environment. Musse used this informed environment to animate human crowds by using a hierarchical control [9]: a virtual human agent belongs to a group that belongs to a crowd, and an agent applies the general behaviours defined at the group level. However, very few works on the modelling of large scale geographic environments using the holonic approach have been found in the literature. Authors in [3] propose a geographic representation of an industrial plant for traffic analysis purposes. This study uses the holonic approach to decompose real world environment in terms of holons. It shows the contribution of such an approach in reducing the complexity of real world environments and demonstrates the capability of the holonic approach to organize various geographic features (road, exchange point, segment, link). However, authors provide no information about the methodology they use to build what they call *3D virtual world*. In addition, we found limits to these authors' claims that their approach can be used to model large scale geographic environments. First, the focus of their work is traffic analysis; therefore the model only takes into account the geographic features that are involved in the traffic flow. Second, the surface of the industrial plant is around ($2km^2$) which is a relatively small area to be qualified as a large scale geographic environment.

The lack of efficient models which are able to represent large scale geographic environments while taking into account the geometric, topologic, and semantic characteristics of the space motivated us to propose a novel approach based on a holonic approach which allows us to merge semantic data, to accurately subdivide the space, and to provide an environment holarchy built upon a topological graph. In the following section, we introduce the fundamental geographic concepts involved in this work.

3 Geographic Data and Spatial Subdivision

GIS data are usually available in either *raster* or *vector* formats [10]. The *raster* format subdivides the space into regular square cells, called boxes, associated with space related attributes. This approach generally presents quantitative data whose precision depends on the scale of the representation. In contrast, the *vector* format exactly describes geographic information without constraining geometric shapes and generally associates qualitative data with each shape. Such data are usually exploited in a *VGE* in two ways [11]: *approximative* and *exact* spatial subdivision methods. The *approximative* subdivision can be used to merge multiple semantic data [12], the locations where to store these data being predefined by the grid cells (*Figure 1(b)*). The main drawback of this discrete method is related to a loss in location accuracy, making it difficult to accurately position any information which is not aligned with the subdivision. Another drawback arises when trying to precisely represent large environments using a grid: the number of cells tends to increase dramatically, which makes the environment exploitation very costly. The grid-based method is mainly used for animation purposes because of the fast data access it provides [13]. The second method, called *exact* subdivision, consists in subdividing the environment in convex cells using the vector format as an input. The convex cells can be generated by several algorithms, among which the most popular is the *Constrained Delaunay Triangulation* (*CDT*) [14]. The *CDT* produces triangles while keeping the original geometry segments which are named constraints (*Figure 1(b)*). The first advantage of the exact subdivision method is to preserve the input geometry, allowing to accurately manipulate and visualise the environment at different scales. Another advantage of this approach is that the number of produced cells only depends on the complexity of the input shapes, but not on the environment's size and scale as it is the case with the grid method. The main drawback of this approach is the difficulty to merge multiple semantic


(a) Approximative decomposition by grids considering two resolutions. White boxes are free, grey are obstacles.

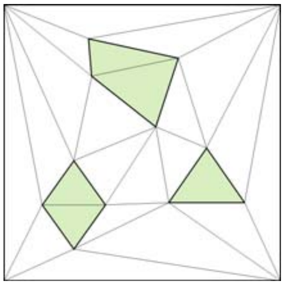

(b) Exact decomposition using *CDT*.

Fig. 1. Two common space decomposition techniques

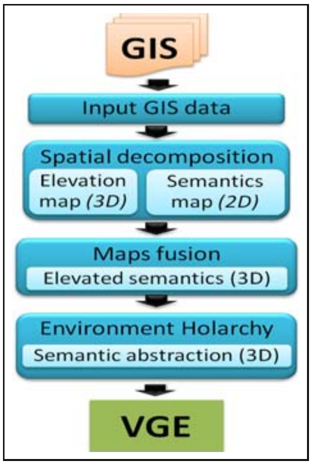
data for overlapping shapes. Moreover, this method is generally used to represent planar environments because the *CDT* can only handle 2D geometries. This method tends to be used for microscopic simulations where accuracy is essential. Two kinds of information can be stored in the description of a *VGE*. Quantitative data are stored as numerical values which are generally used to depict geometric properties (like a path's width of *2* meters) or statistical values (like a density of *2.5* persons per square meter). Qualitative data are introduced as identifiers which can be a reference to an external database or a word with an arbitrary semantic, called a label. Such labels can be used to qualify an area (like a *road* or a *building*) or to interpret a quantitative value (like a *narrow* passage or a *crowded* place). An advantage of interpreting quantitative data is to reduce a potentially infinite set of inputs to a discrete set of values, which is particularly useful to condense information in successive abstraction levels to be used for reasoning purposes.

4 Generation of the Environment Holarchy

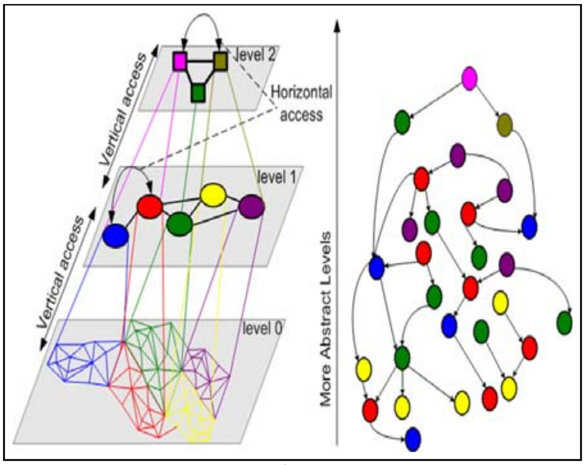
We propose an automated approach to generate the environment holarchy data directly from vectorial *GIS* data. This approach is based on four stages which are detailed in this section (*Figure 2(a)*): *input GIS data*, *spatial decomposition*, *maps fusion*, and *Environment holarchy*. The geographic environment can hence be thought of as a holarchy of abstracted graph in which higher level graphs contain less holons (grouped cells) and have a low discrimination potential in spatial reasoning (*Figure 2(b)*).

4.1 Input GIS Data

The first step of our approach is "***Input GIS data***" and consists in selecting the different *GIS* vector data which will be used to build the *VGE*. The only restrictions concerning these data is that they need to respect the same scale and to be equally geo-referenced. The input data can be organised in two categories. First, *elevation layers* containing geographical marks indicating absolute terrain elevations. Second, *semantic layers* are used to qualify various features of the geographic space. As shown in *Figure 3*, each layer indicates the geographic boundaries of a set of features having identical semantics,



(a)



(b)

Fig. 2. Extraction of the Environment Holarchy: (a) the holonic based VGE generation methodology; (b) the environment holarchy structure

such as roads and buildings. The features' boundaries can overlap between two layers, our model being able to merge this information.

4.2 Spatial Decomposition

The second step of our method is "***Spatial decomposition***" and consists in obtaining an exact spatial decomposition of the input data in cells. This process is entirely automatic, using a Delaunay triangulation, and can be divided into two parts in relation to the previous phase. First, an elevation map is computed, corresponding to the triangulation of the elevation layers. All the elevation points of the layers are injected in a 2D triangulation, the elevation being considered as an additional datum. This process produces an environment subdivision composed of connected triangles as shown in (*Figure 4(a)*) (*Figure 6(a)*). Such a subdivision provides information about coplanar areas: the elevation of any point inside the environment can be deduced thanks to the elevation of the three vertices of the corresponding triangle. Second, a merged semantics map is computed, corresponding to a *Constrained Delaunay Triangulation* (*CDT*) of the semantic layers. Indeed, each segment of a semantic layer is injected as a constraint which keeps track of the original semantic data thanks to an additional datum. Consequently, the resulting map is a *CDT* merging all input semantics: each constraint represents as many semantics as the number of input layers containing it. For example, *Figure 4(b)* and *Figure 6(b)* present the resulting *CDT* of the geographic features provided respectively in *Figure 3* and *Figure 5* by using the same colours for each semantic.

4.3 Maps Fusion

The third step to obtain the *VGE* data is called "***Maps fusion***" and consists in unifying the two maps previously obtained. This phase corresponds to the mapping of the

$2D$ merged semantic map (*Figure 4(b)* and *Figure 6(b)*) on the $2.5D$ elevation map (*Figure 4(a)* and *Figure 6(a)*) in order to obtain the final $2.5D$ elevated merged semantics map (*Figure 4(c)* and *Figure 6(c)*). First, a preprocessing is carried out on the merged semantics map in order to preserve the elevation precision inside the unified map. Indeed, all the points of the elevation map are injected in the merged semantics triangulation, creating new triangles. Then, a second process elevates the merged semantics map. The elevation of each merged semantics point P is computed by retrieving the triangle T of the elevation map whose $2D$ projection contains P. Once T is obtained, the elevation is simply computed by projecting P on the plane defined by T using the Z axis. When P is outside the convex hull of the elevation map, no triangle can be found and the elevation cannot be directly deduced. In this case, we use the average height of the points of the convex hull which are visible from P.

4.4 Environment Holarchy

The fourth and last step of our model is called "***Environment holarchy***". This step is built upon the obtained fused map which contains a set of cells embedding all the semantic information of the input layers, along with the elevation information. Data of this map are mapped to a topological graph, where each node corresponds to the map's triangles, and each arc to the adjacency relations between these triangles. Each node of a graph of level $n+1$ contains a subset of the graph of level n, composed by at least one node. In order to organize this graph-based spatial structure, a holarchy of abstracted graphs is proposed (*Figure 2(b)*). This holarchy is fundamentally based on the following roles: *Head* and *Part*. The head represents members of its holonic organization. Its responsibilities are limited to: 1) integrate new members to its holon group, and 2) to release members which are not relevant to the holarchy. The part responsibilities are basically: 1) remain member of a holarchy, and 2) seek for a holarchy to join. The environment holarchy is obtained by applying a two phase process: 1) *grouping connex cells* to form *groups* of cells, the 2) merging of semantically coherent groups in order to form *zones*. The convex cell grouping phase (phase1) is characterised by a strict geometric constraint considering the convexity of the generated groups. During this phase, the head takes the decision to integrate or realise a member based on the following model:

$$Func_{Head}(h_i) = \left| \begin{array}{ll} \textbf{If } \nabla(Semantic(h_i),\, S) & \textbf{Then} \\ \max(C(Head \cup h_i), C(Head)) & \\ \textbf{Else} & \\ C(Head) & \end{array} \right.$$

$Func_{Head}(h_i)$ refers to the decision making model of the head regarding the integration of the candidate holon h_i. $\nabla(S_i, S_j)$ is a *compatibility* function between semantics S_i and S_j whose evaluation is a *boolean*. C is the convexity factor and is computed as follows: $C(h_i) = surface(h_i)/surface(CH(h_i))$, and $0 < C(h_i) \leq 1$. $CH(h_i)$ is the convex hull of the geometric form corresponding to the holon h_i, $Semantic(h_i)$ is the semantic of the holon h_i, and S is the list of the $Head$ semantics. The $Head$ checks if the semantic information of h_i is *compatible* with its semantic information set S. If the test is conclusive, the $Head$ computes the convexity factor of its geometric form with and without the candidate *holon*$_i$. The integration of the holon h_i must maximise the convexity factor of the $Head$ (i.e. $C(Head \cup h_i) > C(Head)$. The same logic is applied for released members. The release of the holon h_j must optimise the convexity factor of the $Head$ (i.e. $C(Head \setminus h_j) > C(Head))$. The merging of semantically coherent groups (phase 2) is characterised by a less strict geometric constraint regarding the convexity of the generated zones and a higher priority to their semantics compatibility functions ∇. Moreover, depending on the application purposes, designers may apply this second phase grouping strategy several times while progressively increasing the priority of the semantic grouping rules and decreasing the geometric constraint. An example of semantic grouping rules may correspond to the *crossable* relationship between groups. However, designers may freely adopt application-dependent semantic merging rules. For example, a group of cells qualified as *road* can be grouped with a group of cells qualified as *highway* in order to create a zone which enables situated agents representing cars to navigate in the VGE. In the same way, a group of cells qualified as *sidewalk* can be grouped with a group of cells qualified as *crosswalk* in order to allow situated agents representing pedestrians that can cross roads and thus navigate in the VGE.

5 Results

In order to highlight the generic aspect of our holonic approach to model large scale geographic environments, two different kinds of environments have been selected. Fist, a urban environment corresponding to a part of Quebec city (*QC, Canada*) which covers an area of $35km^2$. This urban environment is extracted from various GIS data sources *Figure 3*. The first level of the environment holarchy approximately contains $125,000$ triangles (cells). While the second level contains around $77,000$ groups of cells, and finally the third level encompasses $12,000$ zones. Second, a geographic environment representing the *Montmorency* experimental forest (*St.Lawrence Region, Quebec, Canada*) and covering an area of $47km^2$. This natural environment is extracted from various GIS data sources (*Figure 5*). The first level of the environment holarchy approximately contains $178,000$ triangles (cells). While the second level contains around $92,000$ groups of cells, and finally the third level encompasses $27,000$ zones. The performances of the environment extraction process are very good, being able to process the urban

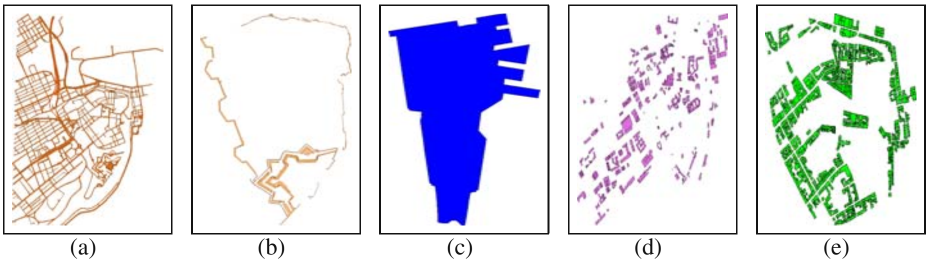


Fig. 3. Various semantic layers related to Quebec City (Canada): (a) road network; (b) old city wall; (c) marina; (d) governmental buildings; (e) houses

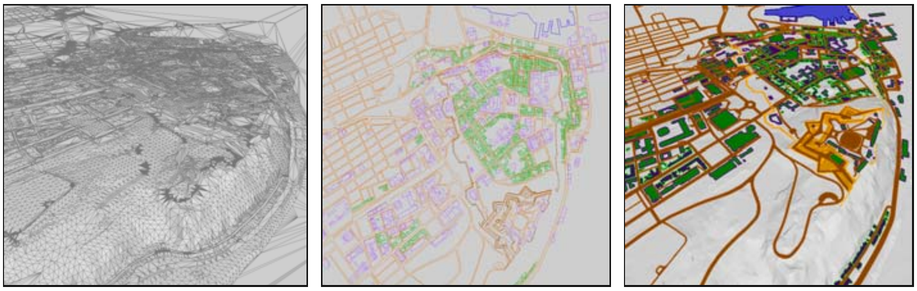

(a) Triangulated elevation (3*D*). (b) Merged semantics (2*D*). (c) Fused map (3*D*).

Fig. 4. Spatial decomposition and semantic merging(*a*, *b*) and the fused map (*c*) of Quebec City

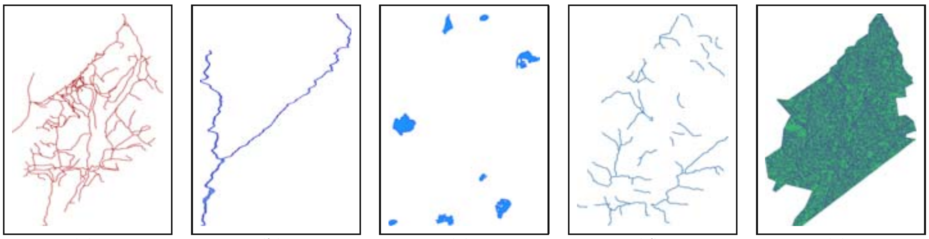

(a) (b) (c) (d) (e)

Fig. 5. Various semantic layers related to Montmorency experimental forest: (a) pedestrian walkway network ; (b) river; (c) lakes, (d) water streams; (e) vegetation

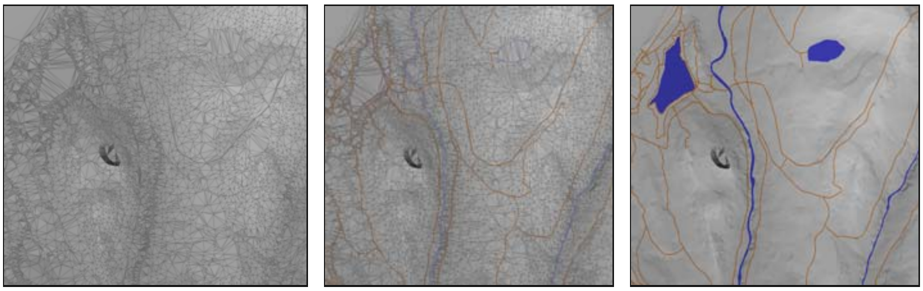

(a) Triangulated elevation (3*D*). (b) Merged semantics (2*D*). (c) Fused map (3*D*).

Fig. 6. Spatial decomposition and semantic merging(*a*, *b*) and the fused map (*c*) of the *Montmrency* forest

environment holarchy in less than 3.2 seconds and the natural environment holarchy in less than 3.9 seconds on a standard computer (Intel Core 2 Duo processor 2.13Ghz, 1G RAM).

6 Conclusion and Perspectives

In this paper, we proposed a novel holonic approach to model large scale geographic environments. This approach extracts an environment holarchy from realistic GIS data and combines the grid-based facility to merge geographic semantic information with the accuracy of vector-based geometric representations. The interest of a holonic view of the environment is that it provides a scalable multilevel model to express complex real world environments. The holonic approach opens perspectives to represent different levels of detail, from a high-level coarse-grained view of the environment to a low-level fine-grained one. In addition, we have shown the suitability of the holonic approach to organise the data generated by the exact space decomposition into an environment holarchy. The environment holarchy allows us to take advantage of common and efficient graph theory algorithms to examine its structure, and especially graph traversal ones. Interacting with the environment holarchy consists to retrieve nodes which correspond to cells, groups of cells, or zones; depending on the holarchy level. Once a node is obtained, it is possible to extract its corresponding data such as the elevation and the semantics information.

Given the above-mentioned characteristics, the environment holarchy opens large perspectives to many spatial reasoning algorithms that can be easily applied, such as path planning or virtual navigation. Actually, we are currently working on an industrial application of the VGE for sensor webs management [6]. Indeed, this informed environment is particularly well suited to such a domain, since it allows efficient path planning, navigation, and deployment algorithms and provides useful spatial data which are needed for situated sensor agents' behaviours.

Acknowledgement

This research has been supported by the *Canadian Defence* (DRDC Valcartier), the *Canadian Network of Centres of Excellence in Geomatics* (GEOIDE), and the *Natural Sciences and Engineering Research Council of Canada* (NSERC). M. Mekni benefited from a PhD scholarship granted by *Fonds Québécois de la Recherche sur la Nature et les Technologies* (FQRNT).

References

1. Weyns, D., Omicini, A., Odell, J.: Environment, first-order abstraction in multiagent systems. Journal of Autonomous Agents and Multiagent Systems 2006, 5–30 (2007)
2. Yang, W., Gong, J., Hu, C., Wang, W.: Collaborative 3d modeling of large-scale virtual geographic environment. In: Technologies for E-Learning and Digital Entertainment, pp. 829–839 (2006)

3. Rodriguez, S., Hilaire, V., Galland, S., Koukam, A.: Holonic modeling of environments for situated multi-agent systems. In: Environments for Multi-Agent Systems II, pp. 18–31 (2006)
4. Fletcher, M., Hughes, J.: Technology and policy challenges to be met for introducing holons into factory automation environments. In: IEEE 2006 (2006)
5. McGuire, P., Liggins, G., Wojcik, P., Benaskeur, A., Brennan, R.W.: The application of holonic control to tactical sensor management. In: IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications, Washington, DC, pp. 225–230. IEEE Computer Society Press, Los Alamitos (2006)
6. Mekni, M., Jabeur, N., Moulin, B.: A generic model for situated holonic multi-agent systems: A case study on sensor web management. In: Torra, V., Narukawa, Y. (eds.) MDAI 2008. LNCS, vol. 5285, pp. 35–46. Springer, Heidelberg (2008)
7. Koestler, A.: The Ghost in the Machine. Picador/Pan Books Ltd., London (1967)
8. Farenc, N., Boulic, R., Thalmann, D.: An informed environment dedicated to the simulation of virtual humans in urban context. In: Brunet, P., Scopigno, R. (eds.) Computer Graphics Forum (Eurographics 1999), vol. 18(3), pp. 309–318. The Eurographics Association and Blackwell Publishers (1999)
9. Musse, S.R.: Human crowd modelling with various levels of behaviour control. PhD thesis, Lausanne (2000)
10. Wang, X.: Integrating gis, simulation models, and visualization in traffic impact analysis. Computers, Environment and Urban Systems 29(4), 471–496 (2005)
11. Gong, J., Hui, L.: Virtual geographical environments: Concept, design, and applications. In: International Symposium on Digital Earth (ISDE), Beijing, China, November 29 - December 2 1999, pp. 369–375 (1999)
12. Zhu, J., Gong, J., Lin, H., Li, W., Zhang, J., Wu, X.: Spatial analysis services in virtual geographic environment based on grid technologies. MIPPR 2005: Geospatial Information, Data Mining, and Applications 6045(1), 541–550 (2005)
13. Tecchia, F., Loscos, C., Chrysanthou, Y.: Visualizing crowds in real-time. Computer Graphics Forum 21(4), 753–765 (2002)
14. Kallmann, M., Bieri, H., Thalmann, D.: Fully dynamic constrained delaunay triangulations. In: Brunnett, G., Hamann, B., Mueller, H. (eds.) Geometric Modelling for Scientific Visualization, pp. 241–257. Springer, Heidelberg (2003)

Holonic Models for Traffic Control Systems

Calin Ciufudean[*] and Constantin Filote[*]

"Stefan cel Mare" University, University str. 9,
720225 Suceava, Romania
{Calin.Ciufudean,Constantin.Filote}calin@eed.usv.ro,
filote@eed.usv.ro

Abstract. This paper proposes a new time-placed net model for traffic control systems, respectively railway control traffic systems. This model can be interpreted as a holonic one, and contains three modules: Transport Planning Module, Transport Control Module and Priority Control Module. For railway traffic systems we introduce a strategy in a timed-place Petri net model to solve collision and traffic jam problems.

Keywords: Petri nets, planning module, priority module, control module, railway traffic, traffic jam.

1 Introduction

In this paper holonic traffic control models are examined, e.g. production tasks like assembly/disassembly or fork/join and traffic fluency of the vehicles with unreliable machines. The problem of maximizing the throughput is considered by achieving a balance among the tasks scheduling and processing of all machines under certain economical criteria [1, 2]. Due to the fact that complex systems built out of simpler subsystems display a remarkable autonomy and stability in the environment, Koestler realized that all the complex structures and stable processes have a hierarchical structure, like living organisms, social societies and non-living systems [3]. Complex systems evolve in shorter time out of simpler systems, with simpler character shapes between the simple and complex ones [4]. For a whole-parted system Koestler denoted the term "holon", derived from the Greek word *holos* = whole with the suffix –on (like neutron or proton) pointing out the part characteristic [5]. A holon could compile strategy out of its rules, which fits to its intentions, and goals and the interpolation of the environment [6]. Typical holonic applications or models are applied mostly to the production processes, called holonic manufacturing systems (HMSs), and we notice that similar to these approaches is that they are modeled domain specifically [7-9]. A complex holonic structure has a hierarchical order, rules and strategies framed by communication and reaction abilities. Most of the time, the hierarchies are looked at as rigid and inflexible shapes. However, some approaches are not following this like the approaches described in [6], and the one proposed here.

[*] Calin Ciufudean and Constantin Filote are with the Electrical Engineering and Computer Science Department of the "Stefan cel Mare" University.

Our approach uses the modeling power of Petri nets in order to build a holonic system for controlling the railway traffic. Petri nets are useful tools for modeling, analyzing and scheduling of a production system; they can provide accurate models of the procedure relations and concurrent, asynchronous events, e.g. holonic systems. Besides, they are also useful in synthesizing plausible control for discrete event dynamic systems. In this paper, a Petri net model is built to model the detailed behavior of a railway system in order optimize some a priori assigned performance criteria.

Due to the fact that the authors worked for more than 10 years for the Romanian Railway Company, the traffic system exemplified is a railway one, although we mention that the holonic models we propose are capable, with minimal changes, to be implemented in various fields of activities.

The ordinary Petri net model is not sufficient to capture the time-related system nature, and hence, timed-place Petri net have been defined and used. In the timed-place Petri net (TPPN), time is associated only with places and all transition firings are instantaneous; this is the model we selected for simulating a railway system. We notice that this paper made only a qualitative analysis of the railway systems; further papers will perform a quantitative analysis [15, 16]. Markings of the TPPN are deterministic during the evolution of the pertaining firing sequence from the initial marking. So, we can directly use the markings of the TPPN model to genuinely describe the states of the system and all the reachable markings can represent the state space of the modeled system. The outline of the paper is as follows: section 2 presents the main characteristics of the railway traffic control systems; section 3 introduces the main components of the proposed holonic models of railway control traffic systems: the transport planning module, the transport control module, and the priority control module. Section 4 concludes the paper.

2 Railway Traffic Control Systems Characteristics

Larger railroads may have multiple dispatcher offices and even multiple dispatchers for each operating division. Generally, railroads used to have their own set of operating instructions under various conditions. In some cases a rail crew might operate on a foreign carrier's lines, and would be required to also know their rules.

To ease this situation, a number of railroads would use a "consolidated code," or set of common operating instructions that were the same for all of the railroads using those.

One of the most common rules from this consolidated code was Rule 251 [14]:

> Rule 251. On portions of the railroad, and on designated tracks so specified in the timetable, trains will run with reference to other trains in the same direction by block signals whose indications will supersede the superiority of trains.

Centralized traffic control (CTC) is a signalling system used by railroads. The system consists of a centralized train dispatcher office that controls railroad switches in the CTC territory and the signals that railroad engineers must obey in order to keep the traffic moving safely and smoothly across the railroad. In the dispatcher's office there is a graphical depiction of the railroad on which the dispatcher can keep track of trains' locations across the territory that the dispatcher controls.

CTC was designed to enable the train dispatcher to control train movements directly, bypassing local operators and eliminating written train orders. Instead, the train dispatcher could directly see the trains' locations and efficiently control the trains' movement by displaying signals and controlling switches. It was also designed to enhance safety by detecting track occupancy and automatically preventing trains from entering a track against the established flow of traffic.

The basic component of a CTC system is detecting track condition and occupancy. The track at either end of the signal block is electrically insulated, and within the block a small electrical current passes through the track. When a train passes a signal and enters a block, the metal wheels and axle of the train short-circuit the current, which causes a relay associated with the track circuit to itself become de-energized. Additionally, any fault in the rail or failure in the signal system, such as a broken rail, a cut wire, or a power failure, will cause the relay to de-energize. When this relay is de-energized, the system understands the track to be occupied or damaged, and the signals show it as such to prevent a train from proceeding and encountering harm. What made CTC machines different from standard interlocking machines was that the vital interlocking hardware was located at the remote location and the CTC machine only displayed track state and sent commands to the remote locations [10].

A command to display a signal would require the remote interlocking to set the flow of traffic and check for a clear route through the interlocking. If a command could not be carried out due to the interlocking logic, the display would not change on the CTC machine. Key to the concept of CTC is the notion of Traffic Control as it applies to railroads. In single direction "Rule 251" operation, each section of track has a timetable defined flow of traffic.

Trains running against the flow of traffic need to be protected by special procedures usually involving some form of absolute block. Implementing a Rule 251 line is very straight forward as the logic for the signaling system is very simple. A bi-directional rail line also needs to avoid the situation of two trains approaching each other on the same section of track.

A bi-directional Rule 251 type system would allow trains to encounter each other head on, and most trains would end up approaching one another at restricted speed. This case would exist if one or both trains receiv a yellow or "Approach" signal and therefore they assume that the next signal block is unoccupied, when in fact there is traffic approaching head on. While the safety issues can be solved with signal block overlaps and other tricks, one will still end up with trains on the same track requiring one train to back up to the nearest passing point.

These systems are conventionally modeled using the operational researches formalisms (e.g. linear diffrential equations, fluency graphs, etc) that can hardly model the system's resources. Our approach presented in the next section improve these models.

3 Holonic Models of Railway Control Traffic Systems

The timed-place Petri net model contains three major modules: Transport Planning Module, Transport Control Module and Priority Control Module. The three modules, of course, are interacted with each other to undertake the necessary actions in response to the triggering from another [18-20].

3.1 The Transport Planning Module

The purpose of the Transport Planning Module is to model the layout of the transport system. When a vehicle (trolley, wagon, or railway engine) needs to move from the current stop to the next stop, it needs to receive a “ticket” of movement first to know its destination. Then, the vehicle acquires the control right of the next stop to make sure that the stop is free at the moment. If both conditions are satisfied, it can start its movement to the next stop. In the same time, the control right of the current stop will be released to allow another vehicle to use it as a destination or pass-by stop. The Transport Planning Module consists of a number of Elementary Transport Layout Modules, as the one shown in Fig.1.

The notations of places used in Fig.1 are explained as follows: st i, i$\in$N represents the stop at a workstation. A token in st i means that a vehicle stays at stop i; ctrl i, i$\in$N, represents the control right of Stop i. If the place ctrl i is marked, it means that Stop i is freed now and all vehicles are allowed to move to Stop i; otherwise, it means there is a vehicle right at the stop so that no other can move to that stop; mv ij, i,j$\in${0,…,4}, with an arc connecting Stop i and Stop j in the layout directed graph, represents the status of the vehicle movement from Stop i to Stop j. A token in the place mv ij means that a vehicle is currently moving from Stop i to Stop j; tk ij, ij$\in${0,…,4}, with an arc connecting Stop i and Stop j in the layout directed graph represents the “ticket” of the path segment from Stop I to Stop j. If there is a token in the place tk ij, it means that a vehicle wants to move from Stop i to Stop j; mv ok represents the status of completing the vehicle movement along a segment path. A token in the place mv ok means that the vehicle has completed the segment path movement.

For example five workstations as shown in Fig.2 are considered. The process flow is: a convoy of railway trucks leaves the garage line (LG) and is sorted in order to complete job 1, respectively to load/unload the wagons at line number 1 or number 3 and then it follows job 2: to load/unload the sorted wagons at line number 2. The next

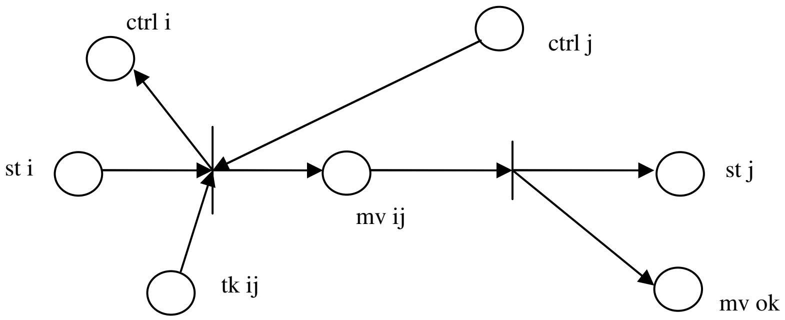


Fig. 1. Elementary Transport Layout Module

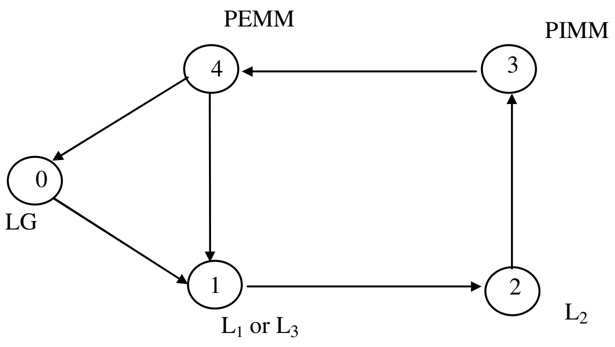


Fig. 2. Layout directed graph

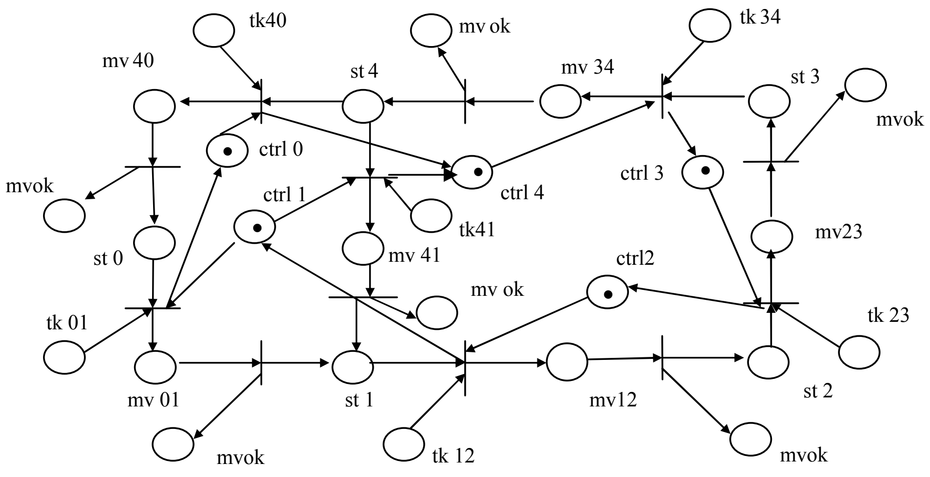


Fig. 3. The Transport Planning Module from Fig.2

operation is to return the wagons to the garage line passing through the expedition lines of the shunt board and to return the flaw wagons to lines 1 or 3. The Transport Planning Module for the layout directed graph in Fig.2 is shown in Fig.3

3.2 The Transport Control Module

This module corresponds to the decision-making Petri net model for vehicle routing. Each time a vehicle wants to move from the current stop to another stop, it must determine its route of movement first. The determined route can be sent to the Transport Planning Module through "ticket" places tk ij to entail the vehicle to move along that route. Therefore, for each stop, we need to have a corresponding Petri net model guiding the traveling path from another stop to it. Fig.4 illustrates the Transport Control Module for a vehicle to move from the current stop to Stop 3. The notations of places different from those in the Transport Planning Module are: mv st i represents the request of a vehicle that wants to move from the current stop to Stop i; at st i means

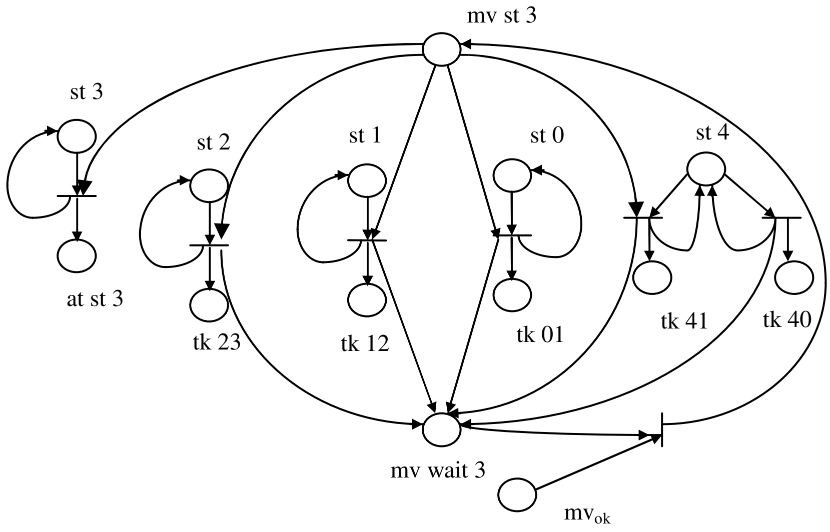


Fig. 4. The Transport Control Module

that the request of a vehicle that wants to move from the current stop to Stop i is fulfilled; mv wait i means waiting for the mv ok condition to generate a "ticket" for the next path segment; mv ok ensures the link with the Transport Planning Module.

3.3 The Priority Control Module

The objective of the Priority Control Module is to introduce a control method into the Petri net model with the aim of guaranteeing jam-free conditions among vehicles. The method is described as follows: when a mission vehicle found a stop on its traveling route was occupied by another vehicle, it will send a command to that vehicle to ask it to leave that stop. After a "ticket" is sent to that vehicle, it starts to move to its next stop and releases the occupancy of its prior residing stop. When the stop is completely released, the mission vehicle can resume its traveling on the same route path. Because the command may be sent to any other vehicle, we must build such models for every pair of vehicles to avoid this kind of traffic problem. The effectiveness of this strategy is limited to the layout geometry and the number of vehicles. But, unfortunately, this is the reality of physical systems [10-14], [21-23].

The Priority Control Model given in Fig.6 is built based on the Elementary Priority Control Model from Fig.5.

To distinguish the places associated with different vehicles, we add the subscript x to a place meaning that the underlying place is used by vehicle x. Consider every three adjacent stop points. If vehicle x is at Stop i and vehicle y is at Stop j and vehicle x wants to move to Stop j while vehicle y is idle, then vehicle y will be "pushed" by vehicle x to the other stop point, e.g., Stop k. Locations mv ok i, i=0, .., 4 ensure the links with the previous discussed modules. So, the Priority Control Module sends a token to the place tk jk y in the Transport Planning Module and, when vehicle y completes its movement, the module will send a token to mv ok y so as to inform vehicle

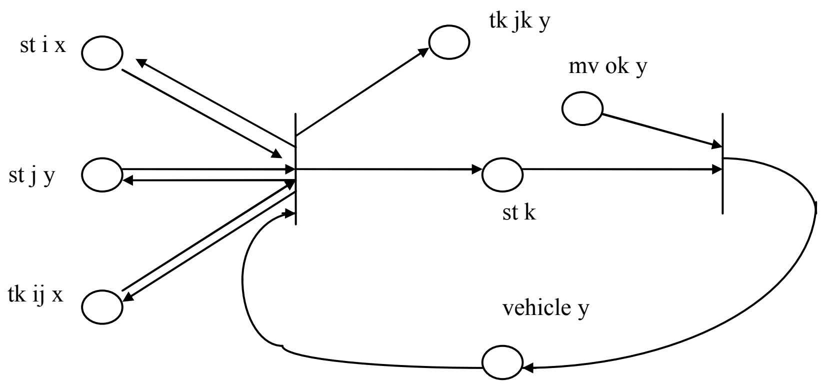


Fig. 5. Elementary Priority Control Module

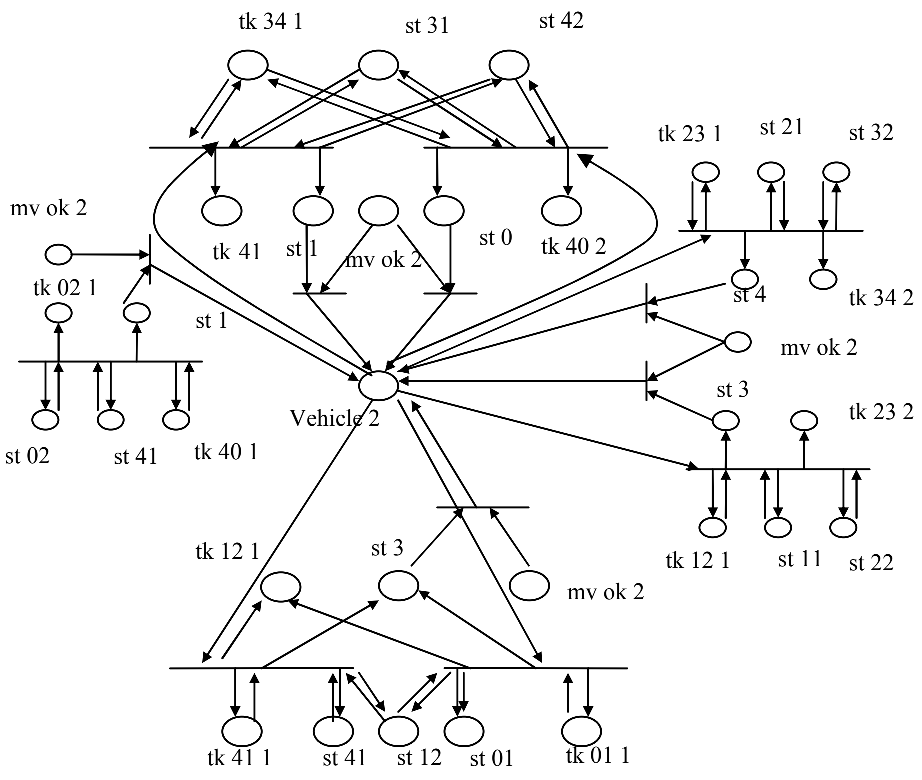


Fig. 6. Priority Control Module

x to start its movement. In Fig. 6 we show the Priority Control Module for the layout directed graph in Fig.2 when vehicle 1 “pushes” vehicle 2.

4 Conclusion

In this paper, we have proposed a new architecture for scheduling and controlling of holonic distributed systems using Petri nets formalisms and their properties [14, 15]. We exemplified this approach with three basic models for railway traffic control system. The proposed approach can be applied to the modeling and analysis of manufacturing systems, supply chains, job scheduling in a chain management system, such as flexible manufacturing systems, etc. The novelty of the approach is that the construction of large Petri nets is not required. Using a structural decomposition, the railway systems are divided into modules. For each module a control structure was derived in order to ensure the traffic safety and fluency. A comparison of the proposed approach with classic railway traffic control formalisms states that our model is less time-consuming, and also more safe, reliable, and versatile than other models. This is due to the fact that Petri net is an intuitive formalism that allows to verify the model by simulating it, and allows also to perform a dynamic change of the model’s parameters (e.g. structure) in order to deliver a safety model that display all the possible events and shows how to avoid the unwanted ones. The objective of the Petri net is to optimize the behavior of a vehicle traveling from the stop at which it currently stays to its destination by ensuring that vehicle collisions and traffic jam are avoided. The efficiency of the proposed method is limited to the layout geometry and the number of vehicles. The presented approach was implemented and tested in our laboratory of discrete event systems and we obtained a real time coordination of the railway system presented above.

The present study may be extended by designing a discrete event controller where there are choices to perform different routes.

Acknowledgments. The research was supported by Romanian National Council for Projects Management within Grant PN II No.71-065/2007.

References

1. Murata, T.: Petri nets: Properties, analysis and applications. Proc. IEEE, 541–580 (1989)
2. Holloway, L.E., Krogh, B.H.: Synthesis of feedback control logic for a class of colored Petri nets. IEEE Trans. Aut. Contr. 35, 514–523 (1999)
3. Koestler, A.: The Ghost in the machine, New York (1976)
4. Fisher, K.: Agent-Based Design of Holonic Manufacturing Systems. Journal of Robotics and Autonomus Systems, Elsevier Science B.V. (1999)
5. Koestler, A.: Jenseits von Atomismus und Holismus – der Begriff des Holons, Wien (1970)
6. Peters, R., Többen, H.: A Reference Model for Holonic Supply Chain Management. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS (LNAI), vol. 3593, pp. 221–232. Springer, Heidelberg (2005)

7. Bussman, S., McFarlane, D.C.: Rationales for Holonic Manufacturing Control. In: Proc. of the 2nd Int. Workshop on Intelligent Manufacturing Systems, Leuven, pp. 177–184 (1999)
8. Delfmann, W., Albers, S.: Supply Chain Management in the Global Context, Koln (2000)
9. Lackmann, F., Nayabi, K., Hieber, R.: Supply Chain Management Software, Planungssysteme im Uberblick, Germany (2003)
10. Ciufudean, C.: Discrete Event Systems for Modelling the Railway Traffic. Matrix Rom Publishing House, Bucharest (2002)
11. Ciufudean, C., Popescu, D.: Modelling Digital Signal Perturbations with Stochastic Petri Nets. Advances in Electrical and Computer Engineering 4(11)(1(21)), 71–75 (2004)
12. Ciufudean, C., Filote, C., Amarandei, D.: Control Charts of Workflows. In: Advances in Data Mning, 8th Industrial Conference, Leipzig, pp. 330–344 (2008)
13. Ciufudean, C.: Artificial Social Systems for Workflow Chart. University of Santander, 8th WSEAS Int. Conf. on Simulation, Modelling And Optimization, Spain (2008)
14. The Consolidated Code of Operating Rules, pp. 133–136 (2008), `http://www.mrcd.org`
15. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Tewneketzis, D.: Diagnosability of Discrete Event Sytems. IEEE Trans. on Autom. Control 40(9), 1555–1575 (1995)
16. Jalcin, A., Boucher, T.O.: An architecture for Flexible Manufacturing Cells with alternate machining and alternate sequencing. IEEE Trans. Robot. Automat. 15(6), 1126–1130 (1999)
17. Recalde, L., Teruel, E., Silva, M.: Modeling and analysis of sequential processes that cooperate through buffers. IEEE Trans. on Rob. and Autom. 14(2), 267–277 (1998)
18. Proth, J.M., Xie, X.L.: Petri Nets. A Tool for Design and Management of Manufacturing Systems. Wiley, New York (1996)
19. Minis, I., Proth, J.M.: Production management in a Petri net environment. Recherche Operationelle 29(3), 321–352 (1995)
20. Zuo, M.J., Liu, B., Murthy, D.N.P.: Replacement-repair policy for multi-state deteriorating products under warranty. European Journal of Operational Research (123), 519–530 (2000)
21. Hopkins, M.: Strategies for determining causes of events, Technical Report R-306, UCLA Cognitive Systems Laboratory (2002)
22. Hall, K.H., Staron, R.J., Vrba, P.: Experience with Holonic and Agent-Based Control Systems and Their Adoption by Industry. In: Mařík, et al, pp. 1–10 (2005)
23. Brusey, J., McFarlane, D.: Designing Communication Protocols for Holonic Control Devices Using Elementary Nets. In: Mařík, et al, pp. 76–86 (2005)

A Multi-Agent System for the Pay-As-You-GO (PAYGO) Social Security Scheme

Athanasios A. Pantelous[1,2] and Alexandros A. Zimbidis[1]

[1] Athens University of Economics and Business, Department of Statistics, 76, Patission Str., Athens, GR-10434, Greece
[2] University of Liverpool, Department of Mathematical Sciences, Mathematics and Oceanography Building, Peach Str., Liverpool, L69 7ZL, U.K.
A.Pantelous@liverpool.ac.uk, {apantel,aaz}@aueb.gr

Abstract. Multi-Agent Systems (MAS) are suitable for dealing with applications where the environment is both dynamic and very complex, with several type of competitors to get involved. Thus, in this paper, we import MAS conceptualization into the well-known (quasi) Pay-As-You-GO social security scheme; practically useful for many Western Economies. First, we start our analysis with the individual agent and subsequently we move towards a system containing many (cognitive) social agents that considers relations, coordinations and interactions.

Keywords: Multi-Agent Systems; Pay-As-You-Go Social Security Scheme; Social Insurance; Contingency Funds.

1 Introduction - The Pay-As-You-Go Social Security Scheme

1.1 The Traditional Version of the Scheme

In this section, some introductive and fundamental elements for social insurance are presented and briefly discussed. This effort is significant, since it provides an overview of the environment in where the agents have to deal with.

The **P**ay-**A**s-**Y**ou-**GO** (**PAYGO**) social security scheme, in its traditional form, requires no accumulation of funds as the other actuarial funding methods do. In practice, there exists a small fund, but only for liquidity purposes. In a certain time period (normally one calendar year), the insured group of lives is divided in two subgroups, the workers (or active members) and the beneficiaries (or pensioners or retirees). According to this approach, the output of today's work-force is partially transferred to today's retirees. Thus, the equation which represents this transition is the following

$$bP = cwL \tag{1}$$

where b is the level of average pension and P the number of eligible beneficiaries (or pensioners). Consumption on pensions, bP, is financed by an appropriate proportional contribution c (percentage rate 100s) on total covered wages (salaries).

Typically, there is an individual wage ceiling above which, no contribution is collected, and normally a maximum pension is associated with. If L is the number of workers who participate in the scheme (not necessarily all the working-age population, see for more details 2^{nd} section) and w is the average covered wage (salary), then cwL is the total respective revenue collected. This simple model shifts immediately any balance perturbation onto beneficiaries or/and contributors.

1.2 The Basic Principles

The concepts of intergenerational solidarity and equity, and the long-term sustainability are the three most important requirements for a successfully operating PAYGO.

The willingness of both young and old generations to participate in a common pool, sharing actual experience, including any losses emerging is defined as **intergenerational solidarity**; see [17], for further discussion.

Moreover, the fact that all the generations are equal to each other is defined as **intergenerational equity**. A metric that is implied this equity, according to [15], is the rate of return. *The interest rate that equalizes the stream of contributions to the interest rate applicable had the contributions actually be invested*, see [9]. This rate of return can be calculated either for individuals or for cohorts of lives.

Furthermore, as the following two paragraphs clearly present, the operating system should also be concerned for the **long-term sustainability** of public pension arrangements under conditions of population ageing.

Member states should undertake ambitious reforms of pension systems in order to contain pressures on public finances, to place pension systems on a sound financial footing and ensure a fair intergenerational balance, see [3].

A strategy pursuing the aim of equal treatment for all generations can cope with . . . demographic changes only through an intergenerational redistribution i.e. a rebalancing in favour of the younger and as yet unborn cohorts, see [13].

1.3 The Rising Cost

Applying the international demographic trends of ageing populations, it is clear that the public pension systems of all the countries operating under the PAYGO scheme will face the rising cost over the next few years. For instance, eq. (1), see [11] and [12], demonstrates that, to maintain balance under a constant amount of a pension benefit, c will have to increase if the decline in fertility rates leads to a decrease in L at the same time as a decrease in mortality rates leads to an increase in P.

1.4 Why Defend the PAYGO Social Security Scheme?

Now, even if there is a considerable rising cost which, will cause a lot of problems in the next decades, the governments still have to defend the PAYGO scheme and the reasons for this choice are very distinct and clear. Many countries have a PAYGO state pension, so a potential change towards a fully funding method will generate a double cost for a certain cohort of lives and an analogous deficit in the national budget till the whole system returns to a new equilibrium under the new financing method.

Additionally, if not all, of these countries have ageing population and therefore need to amend the parameters of the system (such as retirement age, quantum in relation to earnings, number of years of contributory history before qualifying for full benefit, etc) in order to maintain a financially sustainable system. On the other hand, PAYGO scheme can not be useless because there are certain advantages as: the simplicity of its structure, the whole population may be easily covered without any complications, there are no problems with the indexation of benefits or with the efficient investment management as there is no reserve funds etc. (for further discussion, see [1] and [6]).

Consequently, it should be stressed that only considering the reasons above, they provide us with strong evidences why the PAYGO scheme should be further studied and enriched. Definitely, this study is a step into this direction, see next subsections. However, more research work should be done.

1.5 Revisiting the Idea of the Contingency Fund

In [7], it has been firstly introduced a (quasi) PAYGO scheme which incorporates a contigency fund in the system. With the existence of this fund, an interesting combination of the advantages of the PAYGO scheme with those of the fully fund schemes is provided. Moreover, we should mention that this non-zero reserve fund is acting as a buffer, fluctuating deliberately (in the short run) and absorbing partially or completely the unexpectedness in mortality, fertility rates or other random events, see for more details the following section. Afterwards, the contingency fund returns to zero when the fluctuations disappear, leaving the system at a new equilibrium point.

A new extension of the proposed a (quasi) PAYGO scheme by [7] that also adopts the concept of the contingency fund has been considered by Pantelous and Zimbidis, see [11] and [12]. In those papers, several other parameters have been introduced optimizing the derived benefits for the society. For instance, we might mention the different kind of beneficiaries, of contributors etc.

Now, in the end of the 1^{st} Section, we have a clear view for the importance of the PAYGO scheme into the national and global pension systems. In the next section, the PAYGO scheme is further benefited by the use of the Multi-Agent Systems' methodology. A complex system of different agents are introduced for the calculation of different significant parameters.

2 A New Approach Based on Multi Agent Systems

2.1 Multi Agent System

The quasi PAYGO social security scheme proposed by [7], and expanded by [11] and [12], can be further benefited using the **M**ulti-**A**gent **S**ystems' (**MAS**) conceptualization. Analytically, in the existing literature, see for instance [14], [18] and [16], it is more than clear that MAS can supply us with all the appropriate tools to study deeper and even (sometimes) wider what the aspects of inputs are that plausibly explain interactive (social) behaviour. Hence, considering more realistic applications, in our case the PAYGO scheme (as well as every other MAS) can be concerned with the *behaviour of a collection of distributed autonomous (heterogeneous) agents aiming at optimizing funds and making stabilized and viable the national pay-as-you-go social security plan.*

In this part of the section, following the thoughts of [8], we propose the real characteristics of a generic PAYGO scheme. Note that these characteristics are also compatible in several MAS.

– Thus, each agent has incomplete information or capabilities for solving the problem. Additionally, each agent has a limited viewpoint.
– There is no global system control.
– Data is decentralized.
– Computation is asynchronous.

Inferred from the above centralized-importance characteristics, we can assume that the main component founded in every Multi-Agent Systems is the autonomous *agent* and its associated *behaviour* in an *environment*. Consequently, the main contribution of the paper is to explain PAYGO scheme's environment starting with the individual agent and subsequently going towards a system of (cognitive) social agents that considers relations, coordination and interaction between agents. As has already been mentioned, the purpose of the paper is to provide, -according to the authors' knowledge- for the first time, a detailed study of the PAYGO scheme through the notion of MAS, which is very significant paradigm for several other **S**ocial **S**ecurity **A**dminstration (**SSA**) projects.

2.2 The Overall Structure of the PAYGO Scheme

In this section, we have two main goals:

1. First, we want to describe briefly the basic characteristics of each agent appeared in the PAYGO scheme proposed by [11] and [12], and
2. simultaneously we consider relations, coordination and interactions among them.

Moreover, this new approach, which is based on the conceptualization of the MAS, is also able to project (or/and to forecast) effectively social security's finances (i.e. trust fund balances) decades into the future. In this direction, two significant things are required, see [2]:

1. a scheme that shows how *key* economic and demographic agents interact with *policy rules* to determine financial flows, and
2. assumptions about the *annual values* for those agents.

In our analysis, we should also have in our mind that the structure of the proposed quasi PAYGO scheme determines how changes in assumptions affect the systems finances and thus how uncertainty about those assumptions results in uncertainty about the financial status of the SSA policies.

Analytically, a PAYGO scheme should rely on *outputs* of about nine primarily important agents, see **figure 1**. These agents are presented briefly below. We consider the **agent** *for the information about the*

- **fertility rate**
- **mortality rate**
- **immigration rate**
- **unemployment rate**
- **incidence** and **termination** of the claims for **D**isability **I**nsurance (**DI**)
- **real wage growth**
- **inflation** - it affects intermediate demographic and economic variables that determine the accumulation of money in the Social Security trust funds
- **interest rate** - the ninth primary input affects the trust funds directly

The nine agents involved in the PAYGO scheme are living in a highly inaccessible environment surrounded with social (unpredictable or partially

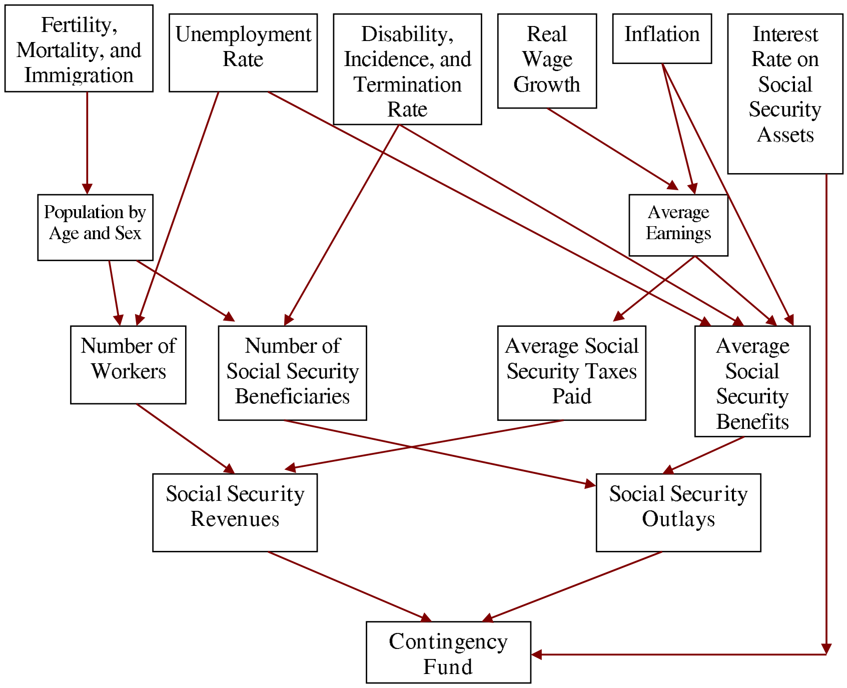

Fertility, Mortality, and Immigration
Unemployment Rate
Disability, Incidence, and Termination Rate
Real Wage Growth
Inflation
Interest Rate on Social Security Assets
Population by Age and Sex
Average Earnings
Number of Workers
Number of Social Security Beneficiaries
Average Social Security Taxes Paid
Average Social Security Benefits
Social Security Revenues
Social Security Outlays
Contingency Fund

Fig. 1. In this figure the nine primary agents that affect the balance of the PAYGO social security scheme are presented. Moreover, their interactions with a line are provided. See [2].

predictable) parameters. Thus, a special language is required to communicate with each other, and not only a simple set of strict rules. This is a consequence, because they continuously have to adapt to new situations in a changing environment. However, in this paper -as we can see from the very next lines- we are not interested in developing any kind of language, since the characterization of the agents' environment and the connection among them are being discussed.

Generally speaking, although the PAYGO scheme is an actuarial model with parameters such as mortality or fertility to be significantly introduced, the interest rate policy is also important, since the higher the interest rate, the faster the trust funds grow (assuming, of course, a positive balance). Moreover, SSA revenues and outlays equal numbers of people (workers or beneficiaries) multiplied by Euro (or dollar)-amounts (average Social Security taxes paid or average benefits). Numbers of people, in turn, are based on population totals, and Euro (or dollar)- amounts are based on earnings.

Those relationships are briefly explained below, see also **figure 1**.

• **Connecting population by age, sex, and marital status:** In practice, every PAYGO scheme begins with a huge multi-column matrix that includes counts of people by *age*, *sex*, and *marital status*. Once the agent has selected future annual values for the mortality and fertility rates and the level of immigration, the model applies the mortality rate to the current population to compute the number of deaths by age and sex. Afterwards, it applies the fertility rate to the female population in order to determine the number of births, for instance, by the age of the mother. This dynamically changing information, along with the assumed net number of immigrants, is added to last years population to obtain the new population figure (with a new age and sex distribution). After that, the agent might distribute the new population among four marital-status groups: single, married,divorced, and widowed - according to age and sex.

• **Calculating the number of workers:** Into the PAYGO scheme, *three* factors are considered to the total working-age population (mainly people ages from 15 to 75) to obtain a view of the total number of workers covered by the SSA, see also [2]:

1. The total working-age population multiplied by the labor force participation rate is finally equal to the *labor force.*
2. The labor force multiplied by the employment rate give the *workers.*
3. The workers multiplied by the covered-worker rate provide the *covered workers labor force participation rate.*

Over the last half century, the labor force participation rate, i.e. the fraction of the working-age population that is employed or looking for employment, has changed significantly. For instance, it is easy to consider that the proportion of women in the labor force has increased steadily and substantially, though the participation rate for women age 65 or older has stayed the same. Although those facts inform complexity, many questions still remain. For example, *how much further will the labor force participation rate for women increase?*, And *long-term trend* *toward earlier retirement for men continue, or will the rate observed over the past 15 years endure?*

• **Measuring the employment rate:** Actually, the employment rate is equal to 1 minus the unemployment rate. In our approach, as well as in several other actuarial models, the unemployment rate is being considered and measured, since it is much more easier to collect data for "unemployed workers".

In practice, the **I**nternational **L**abour **O**rganization (**ILO**) has been regulated that the "unemployed workers" are those who are currently not working, but are willing and are able to work for pay, currently available to work, and have actively searched for work. However, it is clear that not all unemployment is *open* and *counted* by the several government agencies, so the official statistics on unemployment may not be accurate at all. Moreover, into a such complex environment, the agent should have some extra mechanisms that filter out information focusing mainly into the most comprehensive results and enabling the calculation of the unemployment rate by different group categories such as race and gender.

• **Finding the covered - worker rate:** The covered-worker rate is the percentage of employees who work in employment (fully or partially) covered by the SSA. Although that rate will increase slightly as older government workers who are less likely to be covered retire, it is expected to remain relatively stable over the next 75 years, see [2]. Moreover, as it can be easily understood, the covered-worker rate is based on the outputs of the agent which is responsible for the determination of the unemployment rate and, definitely, for several SSA's data.

• **Calculating the number of beneficiaries (or pensioners):** The PAY GO scheme has dozens of categories of beneficiaries, see [10] including retired workers, disabled workers, widows and widowers, and children. Some of the larger categories are broken down by age and sex, but the major division is between retired workers, their dependents, and the survivors of deceased workers and disabled workers and their dependents.

In this case, the agent which is responsible for the outputs relative to the number of beneficiaries, should consider data for each age and sex group. As a result, projecting growth in the number of those beneficiaries is fairly easy for a given age - and sex - specific population. Although changes in labor force participation, earnings patterns, and retirement rates affect the number of beneficiaries, the percentage of the elderly in the population has a much greater impact on Social Securitys finances.

• **Projecting per capita revenue and benefit levels:** Levels of Social Security revenues and benefits per person depend primarily on the growth of wages. That growth in turn can be separated into two sources: the *real wage growth* (which is effectively determined by productivity growth) and the *inflation.*

• **Determining the average revenue levels:** By definition, see [2], the amount of Social Security payroll contributions (or taxes) paid per capita can be determined by multiplying the average effective, contributive (or taxable) payroll by the statutory contribution (or tax) rate. Under current law, the contribution (or tax) rate is constant; thus, the only uncertainty about the average revenue levels comes from the income (or taxable) payroll. Actually, we can use equally the notion "contribution" and "tax", because it is appeared to have the same meaning in PAYGO schemes. Not that there exists a difference between the "tax rate" in Financial and Actuarial Sciences.

In this case, the agent should evaluate firstly the average taxable earnings of the past years, then it might increase that number by considering nominal wage growth as it is clear, in this part of the process, the agent should take into consideration the agents which are providing us with the sum of the inflation and the real wage growth (the first two exogenously projected economic agents).

Growth in the average payroll contribution paid by a worker tracks the average wage growth very closely, but not exactly. Workers pay Social Security contributions only on amounts below the statutory maximum set for taxable earnings. PAYGO scheme does not permit users to vary that assumption. Still, any uncertainty about the effect of distributional changes on the growth of the taxable payroll is probably small compared with uncertainty about the overall wage growth.

• **Determining the average benefits:** An average benefit level must be projected for each of the Social Securitys many beneficiary categories. To analyze the effect of changing the complex benefit formula, PAYGO scheme employs a micro-simulation that projects average benefits for newly retired and newly disabled workers. (Micro-simulation involves producing exact calculations of benefits for a simulated sample of the population and aggregating the results from the sample. That type of simulation is necessary because the effects of the policy changes on different workers vary widely depending on factors such as those workers wage levels and years of employment.)

PAYGO scheme should take into account thousands of newly entitled beneficiaries from *Continuous Work History Data*, which includes each workers entire wage history. Using those wage histories, the model calculates a benefit for each worker and then computes averages for newly retired and newly disabled workers. The input variables in PAYGO scheme that affect benefit levels are *wage growth*, *unemployment*, *inflation*, and, to a lesser extent, *disability rates*.

In the end of this subsection, it should be pointed out that the definition of a *complete* agent, i.e. with all the necessary *components*, is always under strong debates and very deep thought. Many components will be discovered, removed or replaced during the designing process of the agent, consulting the knowledge of an agent model designer. In the literature of MAS, we have found a nice bottom-up example with many significant architecture characteristic, see figure 2 in [4] and [5].

Into this figure, we can clearly observe what is the complexity that occurs when we have to build a model considering just few components. Therefore, in order to obtain good outputs from our model, we have to make great efforts in certain parts of the architecture, for instance how the perception has grown towards a specialised topic - fertility, mortality etc. Fortunately, these

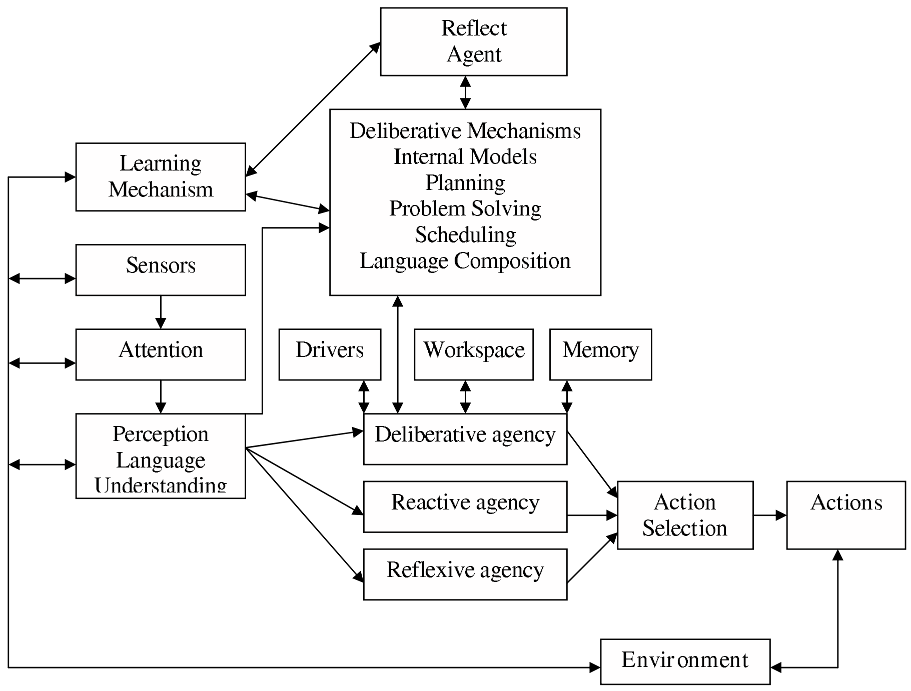


Fig. 2. An agent with components, see [4]

varying points of view have delivered many agent's types that can be put together/constructed from a collection of components; even the way they are implemented varies.

3 Conclusions - Further Research

The system described in this paper presents a new approach to the PAYGO social security scheme proposed by [7], [11] and [12] and has the potential to improve the quality of *decision-making policies* by ensuring more accurate data. Moreover, since several agents have been introduced, this analysis implies that the benefits for the society will be increased.

Different communities of users in the public or/and private actuarial-insurance and financial sector can be identified and used to improve the exploitation of Social Security Administration projects. In this direction, a lot of work should be done. Analytically, we have to provide with more details about each of the agents involved in the PAYGO scheme. Many components of the agent can be discovered, removed or replaced during the designing process. Moreover, when a society has a desire to improve over time the pension system with the help of its experience, then the involved agents require intelligence or a cognitive system's architecture and a memory of representations and experiences from the past in order to be able to learn from previous actions. For the above ideas, further research should be done.

References

1. Brown, R.L.: Pay-As-You-Go Funding Stability: An Age of Eligibility Model. Transactions of the International Congress of Actuaries, 35–56 (1992)
2. Congress of the United States (Congressional Budget Office, CBO),Uncertainty in social security's long-term finances: a stochastic analysis, pp. 1–75 (2001)
3. European Commission. Joint Report from the European Commission and Council on Adequate and Sustainable Pensions. Brussels: Council of the European Union. 6527/2/03 REV2 (2003)
4. Franklin, S.: Autonomous Agents as Embodied AI. Cybernetics and Systems: Special issue on Epistemological Aspects of Embodied AI 28(6), 499–520 (1997)
5. Franklin, S., Graesser, A.: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) ECAI-WS 1996 and ATAL 1996. LNCS, vol. 1193. Springer, Heidelberg (1997)
6. Gillion, C., et al.: Social Security Pensions: Development and Reform. International Labour Office, Geneva (2000)
7. Haberman, S., Zimbidis, A.: An investigation of the Pay-As-You-Go financing method using a contingency fund and optimal control techniques. North American Actuarial Journal 6(2), 60–75 (2002)
8. Jennings, N.R., Woodridge, M.J. (eds.): Agent technologies: Foundation, Applications and Markers. Springer, Heidelberg (1998)
9. Lapkoff, S.F.: A Research Note of Keyfitz's: The Demographic of Unfunded Pensions. European Journal of Population 7, 159–169 (1991)
10. National Statistics Service of Greece (N.S.S.G). Statistical Data for the Development of the Greek Population in 2005-2030, Population Report Series (2004) (The data are available in the relevant website, http://www.statistics.gr)
11. Pantelous, A.A., Zimbidis, A.A.: A Quasi Pay-As-You-Go Financing Model For Controlling The International Demographic Phenomenon of Aging Population. In: Proceedings of the 19th Conference of the Hellenic Statistical Institute, pp. 657–665 (2006)
12. Pantelous, A.A., Zimbidis, A.A.: Dynamic reforming of the traditional pay-as-you-go social security system into a discrete stochastic framework using optimal control methods. Applicationes Mathematicae 35(2), 121–144 (2008)
13. Rupup Commission, Achieving Financial Sustainability for the Social Security System. Berlin, Bundesministerium fur Gesundheit & Soziale Sicherung (Federal Ministry of Health and Social Security) (2003) (English summary)
14. Russell, S.J., Norvig, P.: Artificial intelligence: a Modern approach, 2nd edn. Prentice Hall, Upper Saddle River (2003)
15. Samuelson, P.: An exact consumption-loan model of interest with or without the social contrivance of money. Journal of Political Economy 66, 467–482 (1958)
16. Stone, P.: Layered learning in multi-agent systems: a winning approach to robotic soccer. MIT Press, Cambridge (2000)
17. Wilkie, A.D.: Mutuality and Solidarity: Assessing Risks and Sharing Losses. British Actuarial Journal 3, 985–986 (1997)
18. Wooldridge, M.: An introduction to multiagent systems. John Wiley and Sons Ltd, Chichester (2002)

Contract Monitoring in Agent-Based Systems: Case Study

Jiří Hodík, Jiří Vokřínek, and Michal Jakob

Agent Technology Center
Gerstner Laboratory – Agent Technology Center
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2, 166 27 Praha 6, Czech Republic
{hodik,vokrinek,jakob}@agents.felk.cvut.cz

Abstract. Monitoring of fulfilment of obligations defined by electronic contracts in distributed domains is presented in this paper. A two-level model of contract-based systems and the types of observations needed for contract monitoring are introduced. The observations (inter-agent communication and agents' actions) are collected and processed by the contract observation and analysis pipeline. The presented approach has been utilized in a multi-agent system for electronic contracting in a modular certification testing domain.

Keywords: Contract, Monitoring, Agent, Observation, Case Study.

1 Introduction

The electronic contracting support accelerates a wide variety of electronic business-to-business (B2B) applications. The parties' obligations are captured by electronic contracts, which explicitly specify (semi-)automated actions of contract parties that have to be performed in the electronic or real world under defined conditions. Since the business actors (contract parties) are naturally distributed, autonomous and self-interested, there is a great potential to model them as agents and implement the B2B applications as multi-agent systems.

The concept of electronic contracts in multi-agent systems has been introduced e.g. in [1]. The framework for on-demand service contracts with a high frequency of occurrences is presented in [2]. This work builds on the framework for the electronic contracting presented in [3]. Details of the observation process (in the web services domain) was presented by us in [4].

A key part of a contract-based system is the process through which the actual behaviour of individual agents is checked for conformance with respective governing contracts. Such checking requires that the relevant information on the behaviour of the parties, both with respect to the application processes they execute, and to managing their contractual relationships, is captured. The term *monitoring* has been traditionally used to refer both to the process through which contract-relevant events and states from a running contract-based system are gathered, and the reasoning applied to actually determine their compliance to contracts.

2 Contract

In a general sense, the contract identifies the contract parties and defines the set of restrictions on the behaviour of the contract parties. Contract structure and semantics is described e.g. in [5]. In this work we focus on the desired behaviour defined by a set of clauses, which specify what is expected or allowed to be or not to be performed by the respective contract parties under specified conditions. The workflow derived from contract clauses (alternative paths are allowed) constitutes a possible implementation of the desired behaviour. A deviation from the contracted behaviour usually means a breach of contract clauses and possibly a breach of the contract. The contract may define procedures to solve minor breaches without terminating the contract.

The model of behaviour of contract parties is composed of individual actions, where each action corresponds to an elementary unit of activity. These actions are referenced from the contract, which can prescribe under which circumstances a particular action has to or must not be carried out by a specified contract party. Information about actions performed by respective contract parties is therefore essential for determining adherence of contract parties to respective contracts.

In order to be able to determine contract fulfilment, it is necessary to know two categories of information: (i) information about the behaviour of respective contract parties, and (ii) information about the content and status of the contract [4]. These two categories correspond to a two-level model of contract-based systems, distinguishing between the lower *domain level* and the *contract level* layered on top of it. On the top level, the contract is negotiated between the agents using ACL (agent communication language) messages; on the lower level, the agents influence the environment by performing actions according to the contracted behaviour.

Information about contract-regulated actions is captured at the domain-level in order to determine fulfilment; information about contract-affecting communication between agents is captured at the contract level in order to track which contracts are currently active (and against which the behaviour of contract parties should be checked for fulfilment). For details of the processes and the related type of information, see Figure 1.

Monitoring of the contract execution consists in reasoning above both the data observed at the domain level and action results, which are declaratively described in the contract document by the clauses. For the mapping to the contract clauses the observations are required to hold at least the information about (i) *subjects* to which the observation relates (e.g., ID of agent invoking the action), and (ii) *objects* to which the observation relates (e.g. parameters with which the action is invoked).

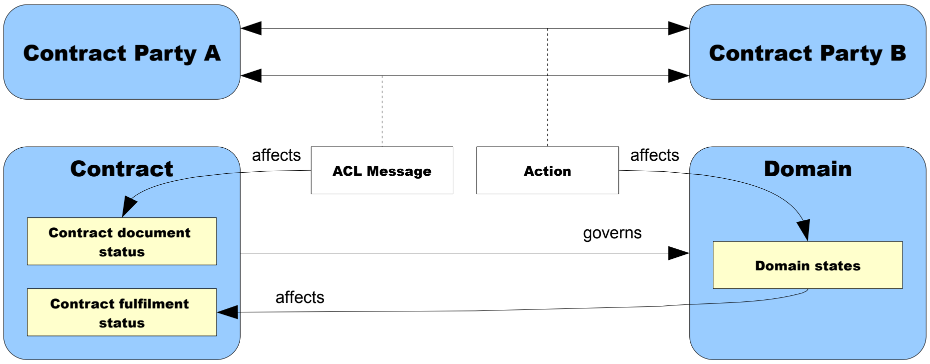


Fig. 1. Observation in contract-based systems

3 Observation and Analysis Pipeline

In the contract environment, data are gathered, collected, and processed by a set of specialized agents forming the *Observation and Analysis Pipeline*. The pipeline consists of (the interactions between the agents are depicted in the Figure 2.):

Sensor, an interface implemented by Contract Parties. The Sensor is responsible for the observation of messages and activities, and reporting them to the Observer. The Sensor may be integrated into the message transport layer to enable automated monitoring, or invoked at Contract Parties' will.

Observer, the central component of the pipeline. The Observer collects data and provides them to the other agents of the pipeline.

Monitor, an optional agent evaluating the data stored by the Observer. For the known contract documents and received reports of the actions performed, the Monitor is able to determine the fulfilment of contracts and their clauses. The outputs of the Monitor are provided back to the Observer and stored there. The Monitor contains an ATN (Augmented Transition Networks) based reasoner [3].

Analyser, an agent dedicated to the evaluation and presentation of the contract-related information obtained from other pipeline components (e.g., Observer) as well as other data sources.

There may also be other agents, which are not components of the pipeline, participating on the pipeline task indirectly. At least one of them (which is crucial for a full and correct operation of the pipeline) is:

Contract Storer, the agent storing the contract agreements and information about their life-cycle state. The Contract Storer agent is residing on top of a database.

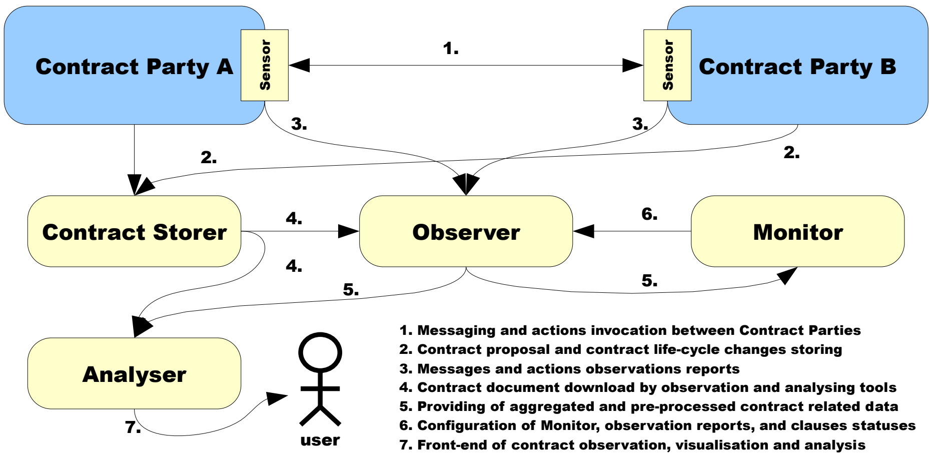

Contract Party A
Sensor
1.
Sensor
Contract Party B
3.
3.
2.
2.
Contract Storer
4.
Observer
6.
Monitor
4.
5.
5.
Analyser
7.
user
1. Messaging and actions invocation between Contract Parties
2. Contract proposal and contract life-cycle changes storing
3. Messages and actions observations reports
4. Contract document download by observation and analysing tools
5. Providing of aggregated and pre-processed contract related data
6. Configuration of Monitor, observation reports, and clauses statuses
7. Front-end of contract observation, visualisation and analysis

Fig. 2. Interactions between the observation pipe-line agents

4 Use Case

The concept of the presented pipeline and the underlying middleware has been validate on several use-cases [6]. This section presents an implementation of the modular certification testing use-case, where the contract defines the testing session and related duties.

The business model of modular certification testing has been developed by Certicon a.s.[1] in cooperation with the Czech Society for Cybernetics and Informatics (CSKI)[2]. Modular certification testing allows a large number of heterogeneous and independent businesses to flexibly collaborate on the provision of certification services. The model has been applied to computer literacy testing using the European Computer Driving Licence (ECDL[3]) concept, first in the Czech Republic and later in Slovakia, and can be equally well applied to other certification programmes of the similar structure.

In line with the use case description, we focus only on the core part of the overall business domain encompassing four types of contract parties (Candidates, Test Centres, Test Room Operators and Testers) and the following contracts:

The Certification Test Contract between the Test Centre and the Candidate (C-TC) gives the Candidate right to attend the test session organized by the Test Centre and specifies terms and conditions for participation of the Candidate on a specific test session organized by the Test Centre and related organization interactions like payment and attendance dues, refunding options and certification granting.

[1] http:/www.certicon.cz/

[2] http://www.cski.cz/

[3] The European Computer Driving Licence is the registered trade mark of The European Computer Driving Licence Foundation Limited in Ireland and other countries. http://www.ecdl.org/

The Test Room Rental Contract between the Test Centre and the Test Room Operator (TC-TRO) gives the Test Centre right to use a test room for the purpose of certification testing and specifies terms and conditions under which the test room is operated and used (e.g., software equipment, number of seats, payment and refunding terms, etc.).

The Tester Hire Contract between the Test Centre and the Tester (TC-T) commits the Tester to provide supervision or marking service to the Test Centre and specifies terms and conditions of this temporary employment relationship like attendance or marking dues, payments, etc.

For a test session to take place, the system requires the instantiation of all the above-mentioned contracts (usually tens of C-TC contracts and several TC-TRO as well as several TC-T contracts, at least one of each type) which are mutually dependent and changes in one of them may affect the other contracts.

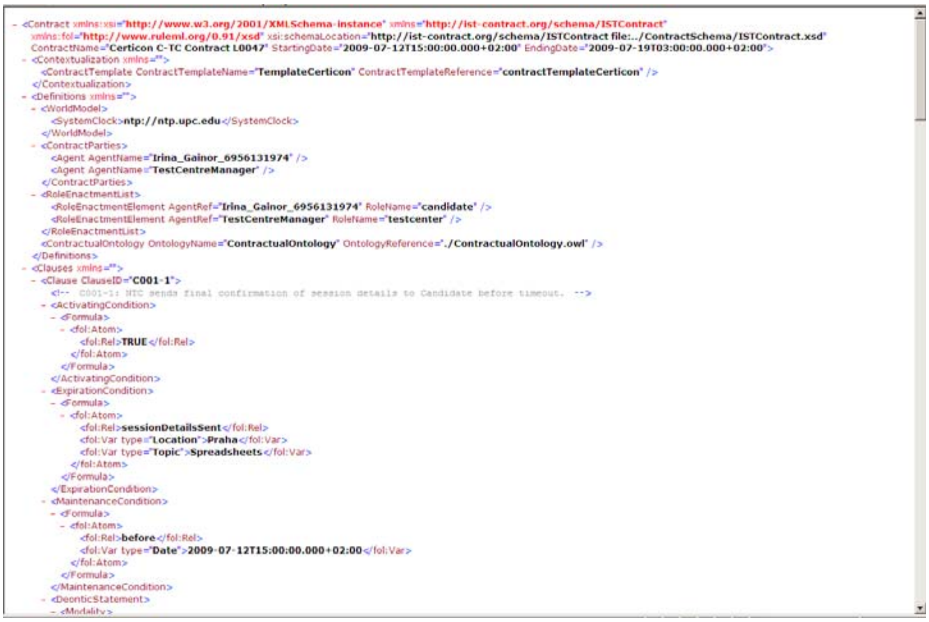
4.1 Usage Scenario

On the top level, the overall scenario can be split into the *contract set-up* phase and the *contract execution* phase. During the contract set-up phase, all the contract parties in the use case interact in order to agree on one or more test sessions. On the consumer side, this involves communication between the Candidates and the Test Center about parameters of requested test sessions; on the supplier side, this involves communication between the Test Center and its suppliers (Test Room Operators and Testers) in order to secure resource required for the intended test sessions. Once the details of the planned test sessions are agreed, the agreement is translated into a set of bilateral contracts which are then signed by the respective parties and enter into force.

In the following, we focus on the contract between the Candidate and the Test Center (the contract parties) and we follow its evolution through the whole contract life-cycle. An example of the contract XML is given in Figure 3.

Contract Creation. The activities in this phase consist of messages exchanged between the Candidate and the Test Center. The flow of activities in this phase is:

1. Candidates send ACL messages to the Test Center with session requests containing a contract template and suggested values for contract variables (e.g., the type of the testing session).
2. The Test Center performs matchmaking between the assembled requests from various Candidates and available suppliers' resources. The best match, i.e. the configuration of suppliers and parameters of the test session meeting restrictions and optimizing preferences of all involved parties, is then translated into a set of bilateral Certification Test contract proposals and propagated back to each Candidate via ACL messages (zero, one or multiple proposals can be send to each Candidate). The contracts are also forwarded to the Contract Storer.



```
- <Contract xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://ist-contract.org/schema/ISTContract"
    xmlns:fol="http://www.ruleml.org/0.91/xsd" xsi:schemaLocation="http://ist-contract.org/schema/ISTContract file:../ContractSchema/ISTContract.xsd"
    ContractName="Certicon C-TC Contract L0047" StartingDate="2009-07-12T15:00:00.000+02:00" EndingDate="2009-07-19T03:00:00.000+02:00">
  - <Contextualization xmlns="">
      <ContractTemplate ContractTemplateName="TemplateCerticon" ContractTemplateReference="contractTemplateCerticon" />
    </Contextualization>
  - <Definitions xmlns="">
    - <WorldModel>
        <SystemClock>ntp://ntp.upc.edu</SystemClock>
      </WorldModel>
    - <ContractParties>
        <Agent AgentName="Irina_Gainor_6956131974" />
        <Agent AgentName="TestCentreManager" />
      </ContractParties>
    - <RoleEnactmentList>
        <RoleEnactmentElement AgentRef="Irina_Gainor_6956131974" RoleName="candidate" />
        <RoleEnactmentElement AgentRef="TestCentreManager" RoleName="testcenter" />
      </RoleEnactmentList>
      <ContractualOntology OntologyName="ContractualOntology" OntologyReference="./ContractualOntology.owl" />
    </Definitions>
  - <Clauses xmlns="">
    - <Clause ClauseID="C001-1">
        <!-- C001-1: NTC sends final confirmation of session details to Candidate before timeout. -->
      - <ActivatingCondition>
        - <Formula>
          - <fol:Atom>
              <fol:Rel>TRUE</fol:Rel>
            </fol:Atom>
          </Formula>
        </ActivatingCondition>
      - <ExpirationCondition>
        - <Formula>
          - <fol:Atom>
              <fol:Rel>sessionDetailsSent</fol:Rel>
              <fol:Var type="Location">Praha</fol:Var>
              <fol:Var type="Topic">Spreadsheets</fol:Var>
            </fol:Atom>
          </Formula>
        </ExpirationCondition>
      - <MaintenanceCondition>
        - <Formula>
          - <fol:Atom>
              <fol:Rel>before</fol:Rel>
              <fol:Var type="Date">2009-07-12T15:00:00.000+02:00</fol:Var>
            </fol:Atom>
          </Formula>
        </MaintenanceCondition>
      - <DeonticStatement>
        - <Modality>
```

Fig. 3. Example of the contract XML

3. Each Candidate evaluates obtained proposals and responds to the Test Center whether the obtained contract proposals are acceptable or not. The contract can be either signed (the contract is concluded and its life-cycle continues by its execution) or rejected (the contract is not accepted and the process terminates). The state of each contract is updated in the Contract Storer.

Contract Execution. The activities in this phase consist of actions invoked by the contract parties. The activities and relations among them are defined in the contract. The flow of states and activities in this phase is:

1. Before session: this state is the initial state of the contract. To move to the next state, following two mutually independent actions must be performed:
   (a) Candidate: pay session fee
   (b) Test Center: provide session information
2. Session, performing test: this state contains activities related to the process of testing. The activities have to be invoked in the following sequence:
   (a) Candidate: register for the session
   (b) Test Center: Provide test to be filled out
   (c) Candidate: Submit test
3. Test evaluation: this state contains activities related to the process of test evaluation and providing the results to Candidate.
   (a) Test Center: notify Candidate about test result.

The actions performed are monitored by the observer (as described in section 3) and reflected in the Contract Storer by update of the contract state. If the flow is successfully finished, the contract life-cycle continues by dissolution phase during which the contract is dissolved. Violations that may occur during the contract execution are:

1. Before session
   (a) Candidate does not pay
   (b) Test Center does not provide session information
2. Session, performing test
   (a) Candidate does not register for the session
   (b) Test Center does not provide a test to be filled out
   (c) Candidate does not submit the test
3. Test evaluation:
   (a) Test Center does not notify the Candidate about the test result

Any of the contract violations terminates the execution phase and start of the dissolution phase with dissolution of the contract. Every action in the contract is safe-guarded by a dead-line whose expiration automatically violates the contract.

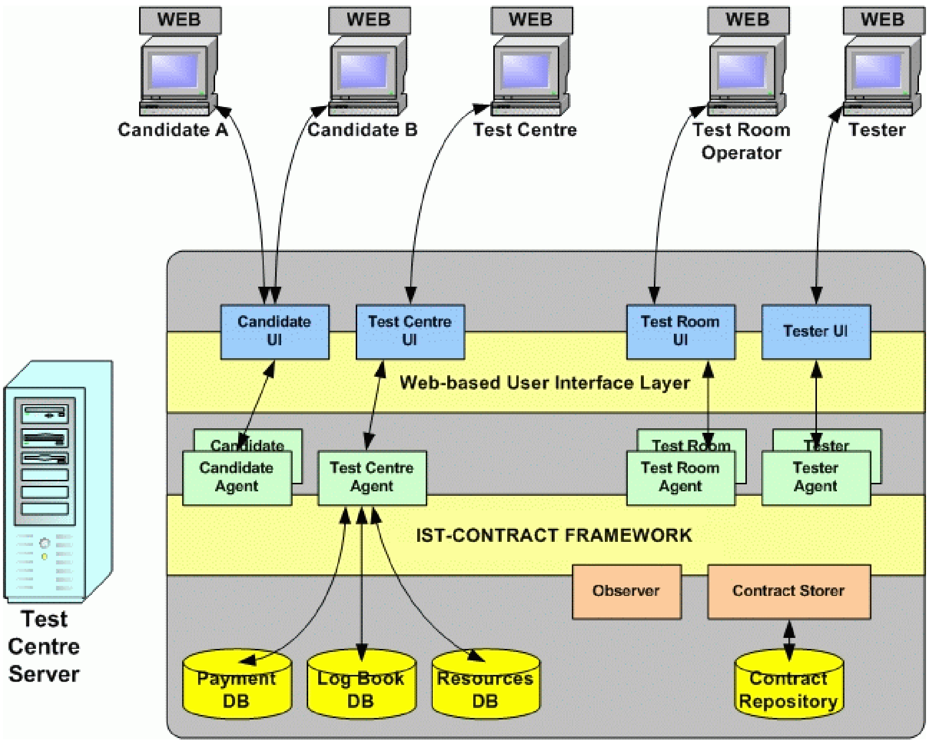
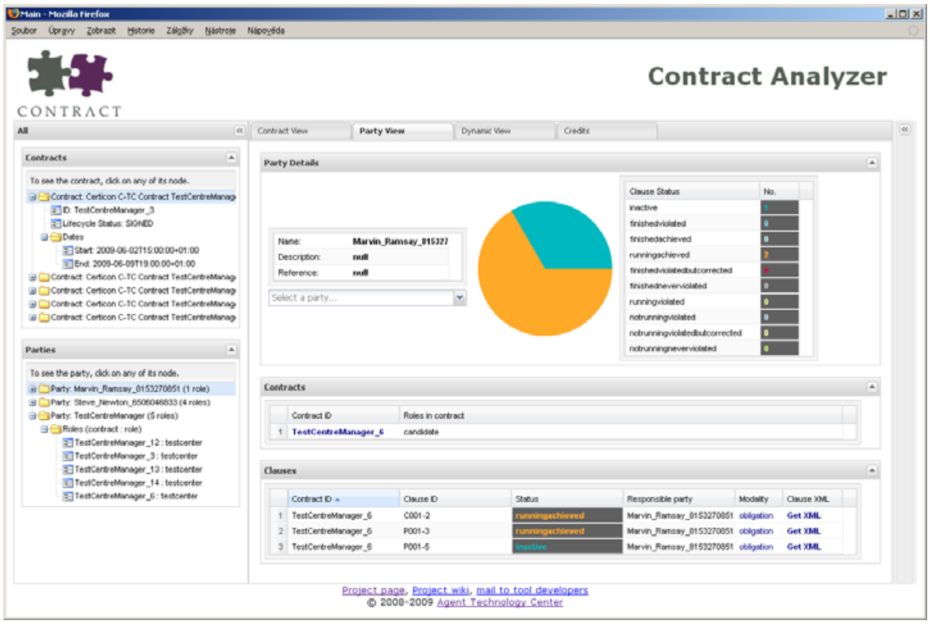


Fig. 4. Prototype architecture



Contract Analyzer

Contracts

To see the contract, click on any of its node.

- Contract: Certicon C-TC Contract TestCentreManag
  - ID: TestCentreManager_3
  - Lifecycle Status: SIGNED
  - Dates
    - Start: 2009-06-02T15:00:00+01:00
    - End: 2009-06-09T19:00:00+01:00
- Contract: Certicon C-TC Contract TestCentreManag
- Contract: Certicon C-TC Contract TestCentreManag
- Contract: Certicon C-TC Contract TestCentreManag
- Contract: Certicon C-TC Contract TestCentreManag

Parties

To see the party, click on any of its node.

- Party: Marvin_Ramsay_8153270851 (1 role)
- Party: Steve_Newton_6506046833 (4 roles)
- Party: TestCentreManager (5 roles)
  - Roles (contract : role)
    - TestCentreManager_12 : testcenter
    - TestCentreManager_3 : testcenter
    - TestCentreManager_13 : testcenter
    - TestCentreManager_14 : testcenter
    - TestCentreManager_6 : testcenter

Party Details

Name:	Marvin_Ramsay_815327
Description:	null
Reference:	null

Select a party...

Clause Status	No.
inactive	1
finishedviolated	0
finishedachieved	0
runningachieved	2
finishedviolatedbutcorrected	0
finishedneverviolated	0
runningviolated	0
notrunningviolated	0
notrunningviolatedbutcorrected	0
notrunningneverviolated	0

Contracts

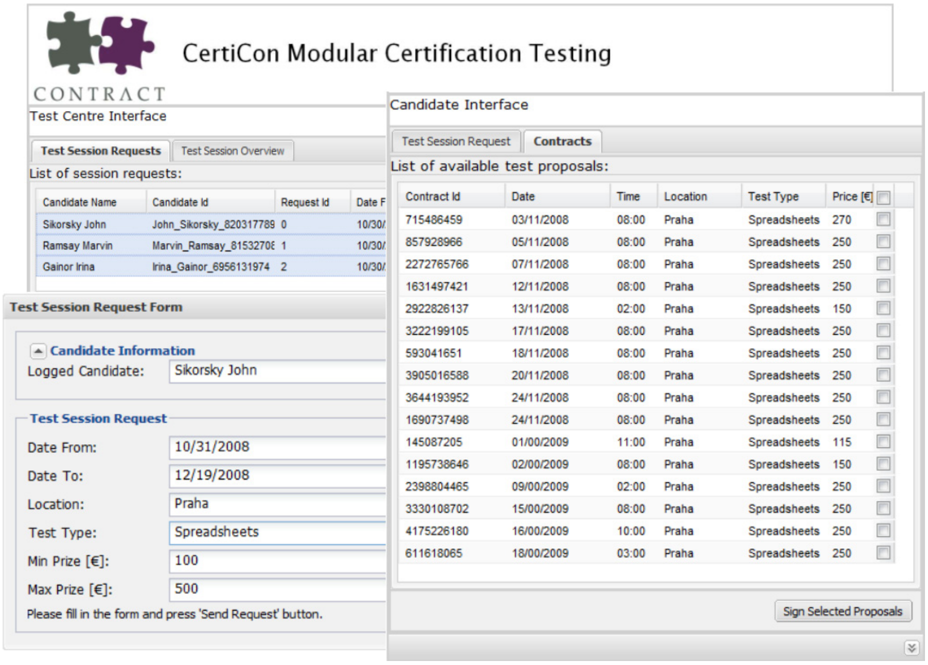
	Contract ID	Roles in contract
1	TestCentreManager_6	candidate

Clauses

	Contract ID	Clause ID	Status	Responsible party	Modality	Clause XML
1	TestCentreManager_6	C001-2	runningachieved	Marvin_Ramsay_8153270851	obligation	Get XML
2	TestCentreManager_6	P001-3	runningachieved	Marvin_Ramsay_8153270851	obligation	Get XML
3	TestCentreManager_6	P001-5	inactive	Marvin_Ramsay_8153270851	obligation	Get XML

Project page, Project wiki, mail to tool developers
© 2008-2009 Agent Technology Center

Fig. 5. Analyser GUI providing the frontend to the Observation pipeline



CertiCon Modular Certification Testing

Test Centre Interface

List of session requests:

Candidate Name	Candidate Id	Request Id	Date F
Sikorsky John	John_Sikorsky_820317789	0	10/30/
Ramsay Marvin	Marvin_Ramsay_8153270E	1	10/30/
Gainor Irina	Irina_Gainor_6956131974	2	10/30/

Test Session Request Form

Candidate Information

Logged Candidate:	Sikorsky John

Test Session Request

Date From:	10/31/2008
Date To:	12/19/2008
Location:	Praha
Test Type:	Spreadsheets
Min Prize [€]:	100
Max Prize [€]:	500

Please fill in the form and press 'Send Request' button.

Candidate Interface

List of available test proposals:

Contract Id	Date	Time	Location	Test Type	Price [€]	☐
715486459	03/11/2008	08:00	Praha	Spreadsheets	270	☐
857928966	05/11/2008	08:00	Praha	Spreadsheets	250	☐
2272765766	07/11/2008	08:00	Praha	Spreadsheets	250	☐
1631497421	12/11/2008	08:00	Praha	Spreadsheets	250	☐
2922826137	13/11/2008	02:00	Praha	Spreadsheets	150	☐
3222199105	17/11/2008	08:00	Praha	Spreadsheets	250	☐
593041651	18/11/2008	08:00	Praha	Spreadsheets	250	☐
3905016588	20/11/2008	08:00	Praha	Spreadsheets	250	☐
3644193952	24/11/2008	08:00	Praha	Spreadsheets	250	☐
1690737498	24/11/2008	08:00	Praha	Spreadsheets	250	☐
145087205	01/00/2009	11:00	Praha	Spreadsheets	115	☐
1195738646	02/00/2009	08:00	Praha	Spreadsheets	150	☐
2398804465	09/00/2009	02:00	Praha	Spreadsheets	250	☐
3330108702	15/00/2009	08:00	Praha	Spreadsheets	250	☐
4175226180	16/00/2009	10:00	Praha	Spreadsheets	250	☐
611618065	18/00/2009	03:00	Praha	Spreadsheets	250	☐

Fig. 6. GUI of the running prototype

Contract Dissolution. In the described case, there are no activities needed for contract dissolution. In the end of the contract life-cycle, the contract status is updated in the Contract Storer and its observation is finished.

4.2 Implementation

At the core of the application is an instance of the IST-CONTRACT[4] framework in which respective contract party agents, one for each contract party in the scenario, are run. Individual actors are represented by agents – a set of Candidate Agents, Test Center Agent, a set of Test Room Agents and a set of Tester Agents (see Figure 4 for an overview of the prototype architecture). Users corresponding to the individual contract parties access the server from other machines via web-based thin clients. For each type of party, there is a separate GUI module supporting the right type of interaction between the entity and the Test Center (see Figure 6).

In addition, the observation pipeline components run on the server, in particular a Contract Storer and an Observer. The Observer is accompanied (not depicted on Figure 4) by the Monitor and Analyser. Those components enable advanced contract monitoring and evaluation. Screenshot of the user interface of the Analyser is on Figure 5. Details about the implementation are presented e.g. in [4].

5 Conclusion

In this paper, we have introduced the implementation of electronic contract-based multi-agent system for the support of modular certification testing. The observation and analysis pipeline has been utilized to enhance the automated contract monitoring and analysis. The system enables fast and efficient monitoring of obligations and their fulfilment that reduces the administrative overhead of the process and speeds up the negotiation of testing sessions.

Acknowledgement

The research described is part-funded by the EC FP6 projects CONTRACT (contract No. 034418), I*PROMS Network of Excellence, and also by the Ministry of Education, Youth and Sports of the Czech Republic grant No. MSM 6840770038.

The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

[4] `http:/www.ist-contract.org/`

References

1. Sallé, M.: Electronic contract framework for contractual agents. In: AI 2002: Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, London, UK, pp. 349–353. Springer, Heidelberg (2002)
2. Streitberger, W.: Framework for the negotiation of electronic contracts in e-business on demand. In: CEC 2005: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, Washington, DC, USA, pp. 370–373. IEEE Computer Society Press, Los Alamitos (2005)
3. Faci, N., Modgil, S., Oren, N., Meneguzzi, F., Miles, S., Luck, M.: Towards a monitoring framework for agent-based contract systems. In: Klusch, M., Pěchouček, M., Polleres, A. (eds.) CIA 2008. LNCS, vol. 5180, pp. 292–305. Springer, Heidelberg (2008)
4. Bíba, J., Hodík, J., Jakob, M.: Contract observation in web services environments. In: Proceedings of Internatonal Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2009), London, UK. Springer, Heidelberg (to appear, 2009)
5. Oren, N., Panagiotidi, S., Vazquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a formalisation of electronic contracting environments. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 156–171. Springer, Heidelberg (2008)
6. Jakob, M., Miles, S., Luck, M., Oren, N., Kollingbaum, M., Holt, C., Vazquez, J., Storms, P., Dehn, M.: Case Studies for Contract-based Systems. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (2008)

A Multi-agent Scheduler for Rent-a-Car Companies

Slava Andreev[1], George Rzevski[2],
Peter Shviekin[1], Peter Skobelev[1], and Igor Yankov[1]

[1] Magenta Technology 349 N-Sadovaya St. Samara 443125 Russia, +7846 342 51 74
skobelev@magenta-technology.ru
[2] Rzevski Solutions, 3 Ashbourne Close W5 3EF London, UK, +44 20 8998 8538
george@rzevski.net

Abstract. The paper gives overview of a multi-agent real-time scheduler for the European operation of one of the largest rent-a-car company in the world. It describes requirements for scheduling of cars and drivers and outlines main features of the ontology-based, multi-agent approach, including systems architecture and performance measurements. The key design decisions and results of the first stage of the system development are also covered. The system is capable of scheduling complex interdependent operations of a large number of resources and of updating schedules affected by the occurrence of unpredictable events in real time. The multi-agent approach developed for scheduling of car rentals can be applied to a variety of complex real-time scheduling and optimization applications.

Keywords: Dynamic scheduling, real-time planning, multi-agent systems, microeconomics, ongoing optimization, events, criteria of decision-making.

1 Introduction

The problem of resource scheduling in real time is one of the most difficult problems in the modern theory of optimization [1-3]. The problem is particularly complex when it contains a large number of interconnected participants and when changes in the schedule of one participant affect schedules of other participants.

In the present paper we consider scheduling of car deliveries for a rent-a-car company, which falls into this class of problems. The paper shows that the problem can be effectively solved using multi-agent technology. Agents, often with conflicting interests, dynamically create a schedule through a process of negotiations and distributed decision-making. The pilot system has been developed and delivered to the client and is working successfully. The second phase scheduler capable of handling 25 rental stations will be delivered in the near future.

The paper covers specifics of a rent-a-car scheduling problem, outlines multi-agent approach to the solution of this problem, describes architecture of the scheduler and singles out main features of its implementation, including performance measurements. The method developed for scheduling rental cars can be also applied to a wide range of other problems in transportation and production logistics, as well as many other industrial problems.

2 The Problem Domain of Rent-a-Car Scheduling

Scheduling problem for a rent-a-car business is deceptively simple: the client specifies the group of the required car, time and place where the car will be picked up and time and place where car will be dropped.

However, the solution to this problem is very complex. The system is required to (a) locate a car belonging to the specified group that can be delivered to the specified location at the specified time, taking into account distances, costs and client's expressed preferences; (b) assign a driver that will wash the car and deliver it to the specified location according to company business processes, taking into account driver's overtime and (c) schedule the transport for the driver to the car location and, after the delivery, to the station where his next job is waiting, or to his home, which may involve another car and another driver. The key difficulty is that the problem is changing while you are attempting to solve it due to the occurrence of unpredictable events: new orders arrive, previous orders are modified or cancelled, cars break down, drivers fall ill or take a day off and delays in delivery occur due to traffic jams.

Usually the territory of a rent-a-car business is divided into a number of small regions each consisting of several rental stations, where a number of skilled managers are engaged in manual scheduling of cars and drivers.

Region-based manual scheduling is however ineffective in respect to the overall optimization of the fleet size, minimization of car delivery distances and achieving acceptable service levels. A more effective scheduling is required for a rent-a-car business to attain sustainable growth and stability.

For developing scheduling solution for rent-a-car business it is necessary to consider the following requirements.

Scheduling of drivers and cars must consider interconnected operations, which are performed at different stages of business processes. A good example is the case when several drivers need to be consolidated in one car to solve delivery and/or collection problems for several orders.

It is important to be able to combine and balance different criteria of decision-making, and to remember that these criteria may change in time. The major criteria include run cost per mile of the car and the driver, the penalty for a delay to rental, the penalty for the late delivery of the returned car to a home station, cost of driver overtimes, and the penalty for not matching the delivered car to the requested car group (upgrade/downgrade).

Managers and drivers in a rent-a-car business are constantly under pressure because of frequent occurrence of unpredictable events and this can negatively affect the quality of schedules of all participants. For example, a car may break down, or the driver could not find keys from the collected car, or could not find the client of the delivered car.

Combined planning and execution is the key feature of a scheduler in the rent-a-car business. The scheduler must respond quickly to a growing gap between the plan and reality, whenever it occurs, because the goal of the system is not only to generate and update schedules but also to monitor and control the execution of planned tasks. The planned tasks are passed into execution by means of mobile devices of drivers who are obliged to confirm the beginning and the end of each operation. A delay in the confirmation that a task has been executed may cause a chain of re-planning of other tasks.

The scheduling process in practice is a process of discovering and solving conflicts among orders rather than a search for the optimal solution. Thus an ability to reach a reasonable trade-off between different criteria is of great importance. To achieve these trade-offs effectively the system makes use of virtual money.

Rent-a-car problems are characterized by the large solution spaces, which make the use of combinatorial algorithms difficult and makes classical methods of local search ineffective. The method described in this paper rapidly responds to unpredictable events as they occur and, in intervals between events, it attempts to find a near-optimal schedule. Designing a scheduler that can cope with such a variety of operating conditions, handle uncertainty related to the occurrence of events and at the same time continuously produce schedules that maximize the set of specified criteria is a real intellectual challenge. To the best of our knowledge such schedulers have not been described in the literature or implemented in practice.

3 Solution

The solution was developed using the multi-agent approach based on the concept of Demand-Resource Networks (DRN) [4-7]. The scheduling process is executed by dynamic interaction of agents representing demands and resources of a company business network. Basic agents for this type of business include: client agents, order agents, station agents, car agents, driver agents, car delivery agents and car collection agents.

The scheduling process starts with partitioning one big task “to deliver a car” into several smaller sub-tasks. Every sub-task is scheduled independently and initiates a complex mechanism of agent negotiations. For example, the agent of a simple pickup task selects the optimum resource (the most suitable car) and creates a subordinated task, say a car wash task, whose agent searches for the driver able to wash the delivered car in specified time. In a more complex case the number of subtasks can be much greater, and tasks have interdependences, i.e., cause and effect relationships. For instance, during scheduling a car rental, the rental agent searches for a suitable car, the car agent searches for the best driver for delivery to the client, the driver agent searches for another driver (runner) to bring him/her back home after car delivery, etc.

The process of matching a resource (a car) to a demand (a rental order) may create or uncover conflicts that will have to be resolved. For example, if during the matching the rental order has selected the car which is already reserved for another client then the conflict can be resolved by refusing this match or by finding another car for the first client, whatever is the nearest to the optimum. Under certain conditions the new rental order can "move" itself to a point in time which, although less desirable, resolves the conflict. Such compromise is required, for example, if all drivers are already allocated for other rentals, and unless the delivery is brought forward, the order cannot be fulfilled. The degree of dissatisfaction of an agent caused by a compromise can be compensated by virtual bonuses. On other occasions agents may be penalized for the infringement of restrictions. Values of bonuses/penalties are determined by the client and can be revised at any time.

A matching between a resource and a demand, established during interaction of agents, is not necessary permanent. Agents perpetually search for improvements to the schedule and may reconsider previously made decisions, break old links and establish new ones, thus reaching new deals that satisfy decision-making criteria better. Pro-activity of agents enables the improvement of the schedule when the system is idle.

In comparison with classical approaches to optimization the multi-agent scheduler described in this paper offers important advantages:

- The scheduler can be easily modified and updated as the business environment and consequently business polices and processes evolve. This can be done by the addition of new agents representing interests of new participants, tasks or resources
- The scheduler is suitable for large scale applications where there is a need to handle thousands of interrelated rentals, tasks and resources, each with a variety of features
- The scheduler performs event-driven dynamic scheduling in real time and is capable of uncovering and resolving conflicts in an existing schedule by changing only affected parts of the schedule
- The scheduler considers every rental order and every resource individually with corresponding agents capable of working with a variety of different criteria, preferences and constraints
- The scheduler allows both users and agents to change criteria, preferences and constraints during its operation and also, if necessary, to modify the balance between various criteria and constraints, e.g., between the service level (car class), delivery time or cost, delivery risks, driver discomfort, etc
- The scheduler is capable of explaining to the users how it made decisions and to offer them opportunities to modify created schedules.

The design of the car rental scheduler is based on earlier designs of schedulers for tankers, trucks and taxi [7-12].

4 Ontology of a Rent-a-Car Business

Ontology is a repository of conceptual knowledge that separates the specific knowledge of the rent-a-car domain from the program code. The problem domain knowledge is represented in ontology as a semantic network of domain concepts, defined by their attributes, and relations. Playing a role of the "dictionary" of the system, ontology provides a basis for constructing instantaneous models of the domain, known as Scenes. A scene shows that at the particular point in time the given car is reserved for the given rental and that it is being delivered to the pick-up station by the given driver. Such a model allows agents to track important links between objects, operations and participants.

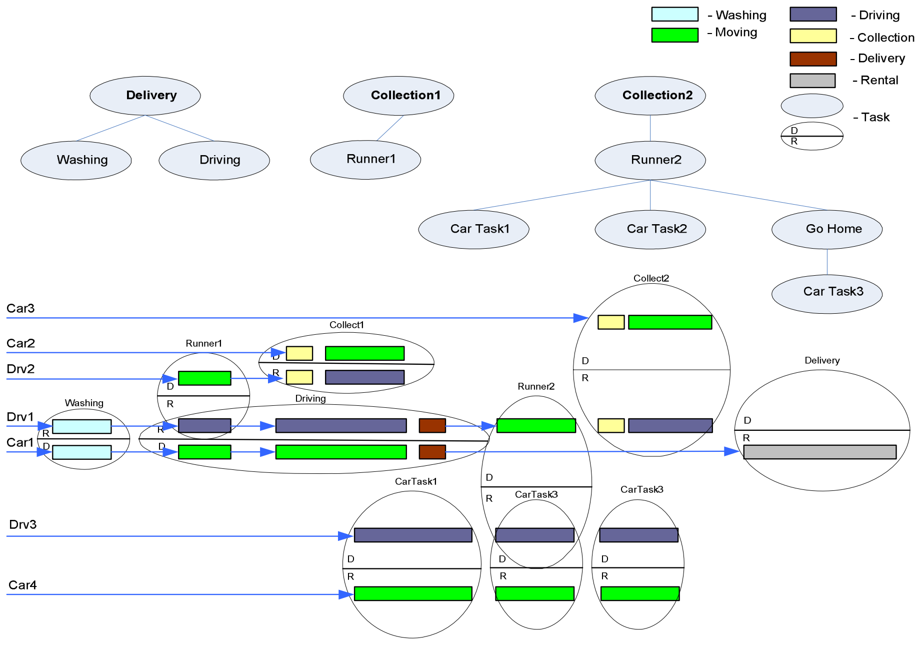


Fig. 1. Example of a Scene

5 Multi-agent World of Rent-a-Car Scheduling

The schedule produced by the multi-agent scheduler is a multi-level network of tasks (operations) linked by relations as shown in figure 1.

Every task and every sub-task have their own agents called Demand Agents. The main goal of a demand agent is to search for the most suitable way of performing the task or sub-task and this includes the interrogation of and negotiation with resources, which are entitled to accept or refuse proposals from demand agents. Agents with different decision-making logic are created for different types of tasks.

A Resource Agent is assigned to every business resource and its first task is to estimate costs of using this particular resource to fulfil a task (rental) proposed by a demand agent. Resource agents calculate preliminary cost (Average Cost) and full cost (Marginal Cost) of using the resource for rental, partition initial tasks into constituent sub-tasks and direct inquiries to corresponding agents. The most important function of a resource agent is to create "promise" (obligation) to carry out a task for a specified sum of money.

The scheduler operates with seven types of tasks, and therefore with seven types of demand agents: Delivery; Washing Task; Driving Task; Collect; Runner Task; Car Task; Go Home Task. It uses two types of resources: Car and Driver.

Agents are driven by Events and majority of events are initiated by demand, for example, scheduling and re-scheduling of a new reservation, change of details of car collection, etc. In some situations resource agents are given power to be proactive and to initiate search for tasks for which the given resource would be the best option. If

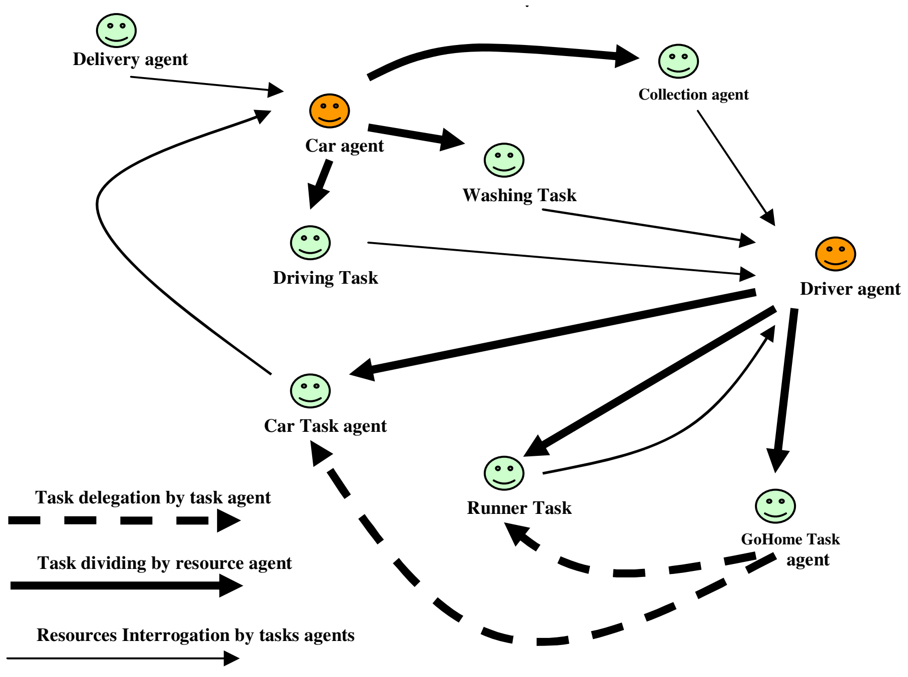


Fig. 2. Multi-Agent World of Scheduling

such a task is found, the resource agent initiates re-scheduling process. Occasionally events are generated by users; these are usually urgent requests for rescheduling due to task failures or changes in schedules of drivers.

The approach to scheduling of car rentals has a number of advantages over other approaches, but it does not guarantee that the created schedule is optimal for all resources for each moment of time, because the scheduler works in real time and the period between the occurrence of events, which cause rescheduling, is too short. The system attempts to create, within the available time interval, a schedule that is as near as practical to the optimum using appropriate heuristics.

To improve the schedule selected agents are temporary made proactive. Perhaps the most important example is the triggering of the Schedule Agent to review the created schedule and report any inefficient use of resources. Using this information, agents identified in the report may attempt to renegotiate their deals. In addition, resource agents from time to time get pro-activity cycles to search for the tasks that may be better suited for them and, if necessary to renegotiate deals with existing tasks. Any renegotiation is permitted only if it improves key indicators for the whole systems. For example, if the driver is waiting in the field, after delivering a car, and is given a new task but his runner is delayed, after a while his agent will start pro-actively searching for a new runner. Thus, in the agent world the pro-activity threads are continuously operating with a view to achieving the optimization of the schedule in a step-wise manner. During re-negotiations the system can temporarily break old links between tasks and resources and thus destroy the integrity of the schedule. As soon as new deals are completed, agents refresh links and recover the schedule, which is now updated.

Ongoing pro-activity is related to KPIs of the schedule. If a current type of pro-activity is not improving KPIs then another type of pro-activity start to push the schedule towards the optimum more aggressively. The approach thus uses “trial and error” method which helps to improve complex schedules and self-regulate internal system activities in a real time.

6 Architecture

The architecture of the multi-regional car rental system is shown in figure 3. The diagram depicts main system components and the 3rd party software.

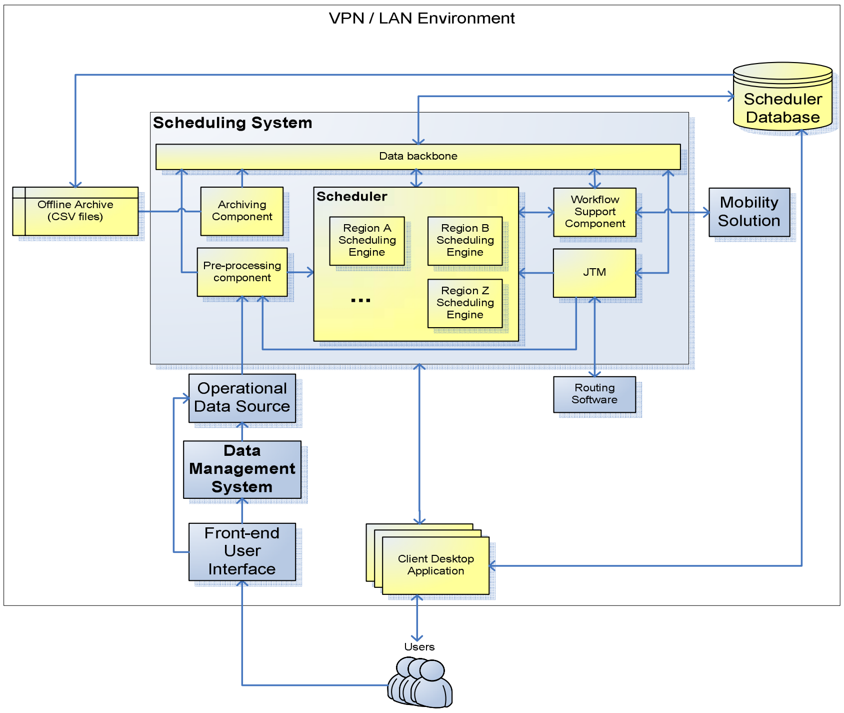
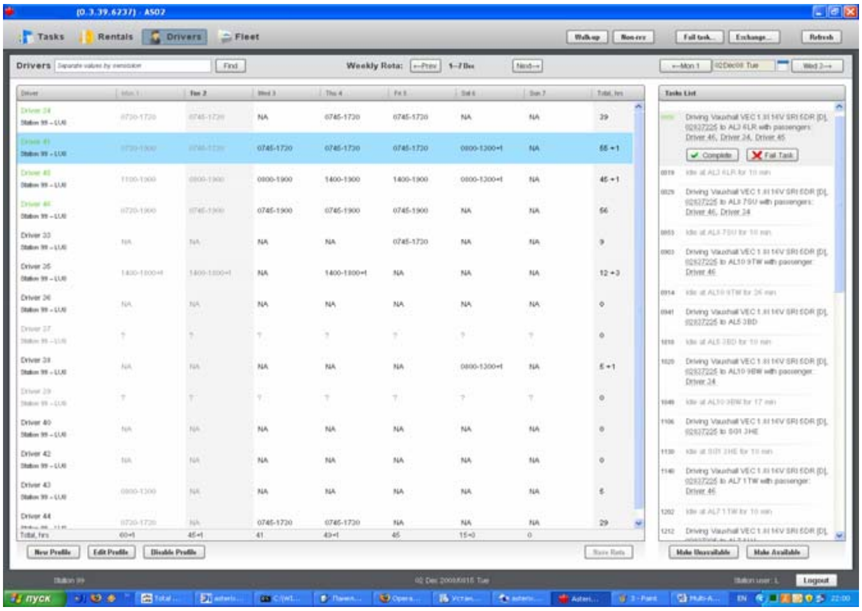


Fig. 3. Architecture

7 User Interfaces

The system provides a comprehensive desktop user interface for day-to-day Station activities (tasks, operational information about cars, drivers and their schedule), Fleet managers (fleet movements functionality) and Administrator (exception management and system maintenance). Client application is automatically deployed and updated on user PCs via Java Web Start service.



Driver	Mon 1	Tue 2	Wed 3	Thu 4	Fri 5	Sat 6	Sun 7	Total, hrs
Driver 24 Station 99 – LU8	0730-1720	0745-1720	NA	0745-1730	0745-1720	NA	NA	39
Driver 41 Station 99 – LU8	0720-1900	0745-1720	0745-1720	0745-1730	0745-1720	0800-1300+1	NA	68 +1
Driver 45 Station 99 – LU8	1100-1900	0800-1900	0800-1900	1400-1900	1400-1900	0800-1300+1	NA	45 +1
Driver 46 Station 99 – LU8	0720-1900	0745-1900	0745-1900	0745-1900	0745-1900	NA	NA	56
Driver 33 Station 99 – LU8	NA	NA	NA	NA	0745-1720	NA	NA	9
Driver 35 Station 99 – LU8	1400-1800+1	1400-1800+1	NA	1400-1800+1	NA	NA	NA	12 +3
Driver 36 Station 99 – LU8	NA	NA	NA	NA	NA	NA	NA	0
Driver 37 Station 99 – LU8	?	?	?	?	?	?	?	0
Driver 38 Station 99 – LU8	NA	NA	NA	NA	NA	0800-1300+1	NA	5 +1
Driver 39 Station 99 – LU8	?	?	?	?	?	?	?	0
Driver 40 Station 99 – LU8	NA	NA	NA	NA	NA	NA	NA	0
Driver 42 Station 99 – LU8	NA	NA	NA	NA	NA	NA	NA	0
Driver 43 Station 99 – LU8	0800-1300	NA	NA	NA	NA	NA	NA	5
Driver 44 Station 99 – LU8	0720-1720	NA	0745-1720	0745-1730	NA	NA	NA	29
Total, hrs	60+1	45+1	41	43+1	45	15+0	0	

Tasks List	
[illegible]	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D], 02937225 to AL3 6LR with passengers: Driver 46, Driver 24, Driver 45
0019	Idle at AL3 6LR for 10 min
0029	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D], 02937225 to AL8 7SU with passengers: Driver 46, Driver 24
0853	Idle at AL8 7SU for 10 min
0903	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D], 02937225 to AL10 9TW with passenger: Driver 46
0914	Idle at AL10 9TW for 26 min
0941	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D], 02937225 to AL5 3BD
1010	Idle at AL5 3BD for 10 min
1020	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D], 02937225 to AL10 9BW with passenger: Driver 24
1049	Idle at AL10 9BW for 17 min
1106	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D], 02937225 to SG1 3HE
1130	Idle at SG1 3HE for 10 min
1140	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D], 02937225 to AL7 1TW with passenger: Driver 46
1202	Idle at AL7 1TW for 10 min
1212	Driving Vauxhall VEC 1.8I 16V SRI 5DR [D],

Fig. 4. Task list

Table 1. Performance matrixes

The amount of cars	250
The amount of drivers	30
Average stream of rentals	15-20 reservations per day for one station
Average number of late task	A driver was late once for 5 minutes for one task, on the average. Approx. 60 % of all tasks were late.
Average frequency of events	About 80-120 events per hour
Time for processing of one event	Maximum - 130 sec, average - 7 sec (70 % of all events were processed in the system in less than 3 sec)
Number of the tasks which are scheduled for one event	From 150 till 200, including: Runner Tasks – 30.7 ,Driving Tasks – 7.28, Washing Tasks – 6.64, Delivery – 1.88, Car Task – 137.89, Collect - 3.75
Messaging intensity in the system	Tasks involving a car, establish 200-250 connections. Tasks involving drivers – 30-40.The number of messages per task is at least 1500
Number of simultaneously active agents	Task agents - 2500 Car agents – 250 Driver agents – 30
The greatest depth of a negotiations chain	10 levels (limited manually)
What percent of decisions is made more than an hour before the beginning of schedule execution	Less than 1 %

The Drivers screen (figure 4) displays all drivers and their working hours planned for the current week (Rota): by weekdays and the total number of hours. To see tasks currently planned for a driver the user selects the driver from the list. Tasks are listed in the chronological order.

8 Performance Matrixes

At present the system is tested on a network of stations of a large rent-a-car company. A number of performance matrixes, which represent most important characteristics of the developed scheduler are presented in table 1.

9 Conclusion

The system described in this paper can be considered as a one of the first industrial multi-agent systems for dynamic scheduling.

The system exhibits a considerable intelligence by being able to autonomously react to events in real time, to generate schedules by resolving many conflicting issues, execute the schedule and monitor results. When differences between the schedule and the reality is recognized the systems triggers re-scheduling processes attempting always to match the plan to reality. When there is sufficient time between two events the system autonomously works on improving given KPIs. Pro-activity and self-organisation are the key system features.

A number of R&D issues have been identified, which need further investigations, including measuring quality of produced schedules, effective interaction with users, balancing conflicting performance criteria and freezing last minute changes, etc. For example, frequent human errors, delays of drivers, incorrect addresses of deliveries and collections, and failed driving tasks can lead to long "ripple effects" of changes in schedules, occasionally too near to the execution.

Our first results prove that multi-agent technology forms a solid basis for addressing above issues and developing new types of solutions, which can provide great opportunities for solving very complex scheduling and optimization problems in real time.

References

1. Leung, J.Y.-T. (ed.): Handbook of Scheduling: Algorithms, Models and Performance Analysis. Computer and Information Science Series. Chapman & Hall / CRC, Boca Raton (2004)
2. Voß, S.: Meta-heuristics: The state of the art. In: Nareyek, A. (ed.) ECAI-WS 2000. LNCS, vol. 2148, p. 1. Springer, Heidelberg (2001)
3. Rego, C., Alidaee, B. (eds.): Metaheuristic Optimization Via Memory and Evolution: Tabu Search and Scatter Search, Boston-London. Operational Research & Computer Science Series (2005)
4. Dorer, K., Calisti, M.: An adaptive solution to dynamic transport optimisation. In: Pechoucek, M., Steiner, D., Thompson, S. (eds.) Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track, pp. 45–51. ACM Press, New York (2005)

5. Vittikh, V.A., Skobelev, P.O.: The multi-agent models of interaction in demand-resource networks. Automatica and Telemechanica (1), 177–185 (2003)
6. van der Putten, S., Robu, V., La Poutré, H., Jorritsma, A., Gal, M.: Automating supply chain negotiations using autonomous agents: a case study in transportation logistics. In: Proceedings of the fifth international joint conference on Autonomous agents and multi-agent systems, Hakodate, Japan, May 08-12 (2006)
7. Rzevski, G.A., Skobelev, P.O., Andreev, V.V.: MagentaToolkit: A set of multi-agent tools for developing adaptive real-time applications. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS, vol. 4659, pp. 303–313. Springer, Heidelberg (2007)
8. Rzevski, G., Himoff, J., Skobelev, P.: Magenta Technology: A Family of Multi-Agent Intelligent Schedulers. In: Proceedings of Workshop on Software Agents in Information Systems and Industrial Applications (SAISIA). - Fraunhofer IITB, Germany (February 2006)
9. Himoff, J., Skobelev, P.O., Wooldridge, M.: Multi-Agent Systems for Ocean Logistics. In: Proceedings of 4-th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), Holland (July 2005)
10. Himoff, J., Rzevski, G.A, Skobelev, P.O.: Magenta Technology: Multi-Agent Logistics i-Scheduler for Road Transportation. In: Proceedings of 5-th International Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2006), Japan (May 2006)
11. Glashenko, A., Ivashenko, A., Rzevski, G., Skobelev, P.: Multi-Agent Real Time Scheduling System for Taxi Companies. In: Decker, Sichman, Sierra, Castelfranchi (eds.) Proc. of 8th Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2009), Budapest, Hungary, May 10–15 (2009) (in publ.)
12. Rzevski, G.A., Skobelev, P.O.: Emergent Intelligence in Large Scale Multi-Agent Systems. Education and Information Technologies Journal 1(2) (2007)

A Framework for Multi Robot Guidance Control

Onur Keskin and Erol Uyar

Dokuz Eylül University Mechatronics Engineering Department, 35100 Bornova İzmir Turkey
{onur.keskin,erol.uyar}@deu.edu.tr

Abstract. In this paper we present a framework for path planning and path finding for multiple mobile robots with global vision. Our framework model considers the agents' dynamic status and their environment with obstacles to perform given tasks. The global vision system provides feedback to main controller computer and mobile robots are directed towards to their tasks with avoiding obstacles and without any collision. Different kinds of scenario are prepared to simulate manipulating tasks and non-collision behavior with our framework. Experiment results with Lego Mindstorms NXT shows that our framework can be used where a multi robot system is needed with minimum resources.

Keywords: multi-agent systems, mobile robot, global vision, path finding.

1 Introduction

Multi Robot Systems have received increased interest in recent years because of their remarkable properties and advantages such as fault tolerance, larger range of task domains, robustness, improved system performance and lower economic cost [8]. Especially interest of multi robot system research is in developing path finding and planning, traffic control, formation control, localization, exploration and foraging. These subjects also studied in the past with single robot systems. But the concept of using multiple robots has many advantages. Multi Robot coordination has the potential to complete the task more efficiently than a single robot. Each mobile robot in Multi Robot Systems can complete different tasks in the same time by working separately which is more reasonable and suitable for the agent [10]. The "agent" word has different meaning in many research areas. In our research topic, "agent" can be defined as a robot system that observers its environment with the help of various sensors and perform action with different types of actuator to achieve assigned task. Algorithms that an agent has and data that is acquired from environment construct agent's program. Agent's architecture encloses sensors and actuators.

Most of the multi robot system has to solve the shortest path problem to achieve given task. This task can be transportation of a load in a factory, or going to destination with using minimum resources. In the last decades path planning and path finding computation methods are discussed, most of new method and algorithm is published [11], [12]. The system must compute possible paths with given properties. Also it is much difficult when the size of agents is increased. We also do not confuse that path finding and path planning are different concepts. But they are related to solve the problem. When path planning process calculates possible paths in the working environment with existent properties and also given limitations and priorities, path finding process consider the capability of agent to execute path that is founded. Path planner has a feedback from path finder to recognize which one is impossible and to be reconsidered.

We develop robust, reliable, programmer friendly agent based framework which uses global vision as a location feedback sensor setup, and assigns various kind of task to multiple robot in the vision space. Our Framework is developed under Microsoft .NET Framework with C# (C Sharp) language. Framework has path planning, vehicle drive and computer vision modules. These modules and used algorithms are defined in the next section. Second, we make two different experiments. The first experiment is to perform a task with only an agent to represent basic capabilities. The second experiment is to show how to handle with complex tasks with multiple agents. Eventually experiment results are discussed and we concluded paper with future works.

2 Framework Description

Our framework is design to solve guidance problem of multi mobile robots. This problem has three general modules to discuss. These modules are global vision, path finding and vehicle tracking [9].

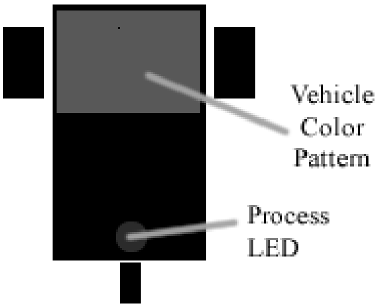
First part is gathering data from environment with only a single overhead camera. The reasons for this usage are to decrease the cost of total system and moreover to share all the data between robots. It is better to manage them from one centralized computer. Centralized computing and managing unit can foresee actions and behave considering the whole working environment. We can express this choice with an everyday example. In a laboratory, students can be distributed into some number of groups or individually to make the experiment, ask their own questions for their own problems to the teacher. Or whole students can make the experiments under teacher's surveillance. This approach comes with a great advantage. The students do not only learn their own problems' solutions, they can learn, or access all the solutions in the laboratory class. And hiring one teacher for each student is never considered as a reasonable option because of cost and redundancy. So we use this principle to supply feedback from our mobile robots with the help of global vision system.

Second part is to finding the shortest path from initial location of a mobile robot to its target location. This step is also combining two sub modules; one of them is used for tracking each mobile robot by the help of global vision system. Second sub module is called "collision checker" which plans the motion step of each mobile robot.

And the last module is used for sending driving parameters and reading mobile robot device information from each mobile robot.

2.1 Computer Vision Module

In this module of framework, we use general computer algorithms to get location of each mobile robot and generate a global map [6]. In first step, each successfully



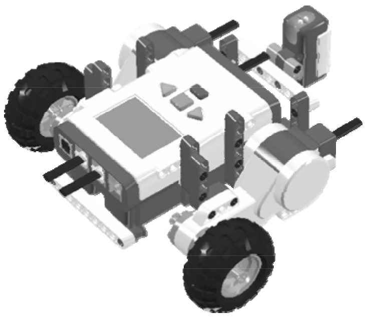


Fig. 1. Schematic top view of mobile robot, process LED color is red and pattern can be defined by framework user/programmer (left), A mobile robot which is designed with "Lego Digital Designer" (right)

connected mobile robot's process LED (figure 1) is triggered until this module gets its pixel location. After getting pixel location, algorithm starts from this coordinate to search in all direction for finding mobile robot's identification pattern (figure 1). In our experiments we use predefined color patterns but other possibilities are going to discuss in future works section. These predefined color pattern names can be written as devices' name to "Lego Mindstorms NXT" modules. And user does not need to check every vehicle in a multi agent situation.

Searching algorithm (figure 2) uses adaptive threshold approach. In this approach the process LED coordinate is taken as center and searching each predefined color, if the neighbor pixels are close enough to one color's threshold value, search is stopped and mobile robot is identified. If no color is matched, the search area is increase until the limit condition. This algorithm can be replaced with connected component analysis, especially blob detection which is used for mobile robot tracking, will suitable for defining mobile robot's identification. Color extraction algorithm is used to perform blob detection more accurately [13]. Color tracking module is directly interacted with user, if any condition is chanced it can easily adjust. Obstacles in environment

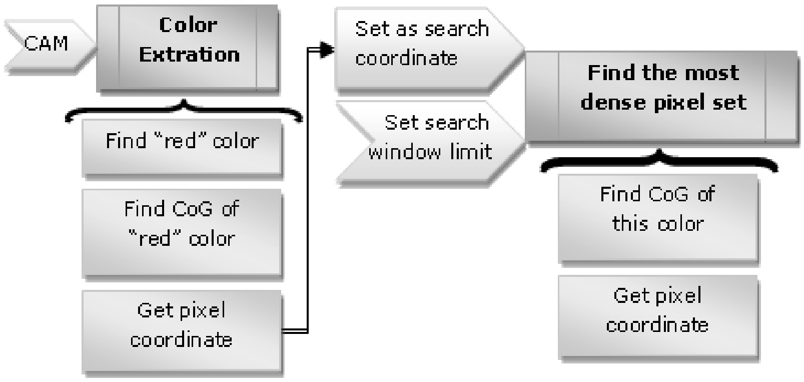


Fig. 2. Flow diagram of finding mobile robot coordinates

are detected with color extraction algorithm. It is assumed that they have a specific color from the environment and from this data; module can generate a global map which can be dynamically change and also saved for further process. Framework also gives an opportunity to create a virtual environment that has obstacles or virtual agents. This property is useful for simulating various combinations without actually have the system.

2.2 Path Finding Module

This module's first and primary task is to solve the shortest path from a starting coordinate to an end coordinate (table 2). Standard path finding algorithms [1], [2], [3], [5] which are used for many application areas are chosen to calculate path.

Our main new approach is to adopt these well know algorithms in a practical and reliable way. This module's point of origin is based on ant behaviors in their community [4]. In multiple robots controlling, it is not sufficient to guide robots from an initial location to target location. Controller must plan what if scenarios to avoid dynamic obstacles like other robots.

This module produces two types of task lists, motion and path, which are manipulated by mobile robots. These lists are recalculated when a new frame is captured from overhead camera with 10 fps resolution. Path list has the coordinates of the path which is generated. Motion list has the movement type to finish the given task. There is also behavioral information in the motion list. This information gives priority to mobile robots. It has two different parameters and they can be chosen by programmer. One of the parameter is to assign priority according to path list in descending order. Second one is to assign priority according to battery level in ascending order. With these priority options, mobile robots can solve stuck on a hallway problem easy as shown in figure 3.

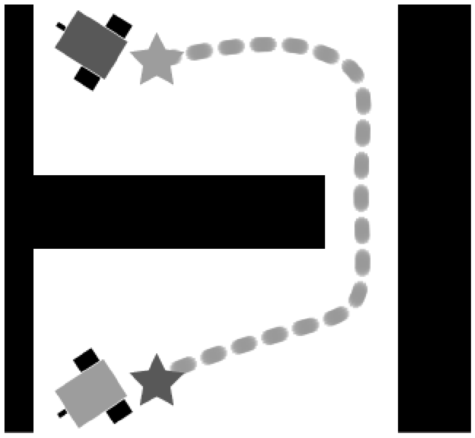


Fig. 3. General stuck on a hallway problem when controlling multiple mobile robots

2.3 Vehicle Drive Module

This module sets power status for priority computation, activates process LED for vehicle identification and matching and controls each mobile robot with getting motion list from path finding module. In order to get and set these data, virtual communication ports must create from Bluetooth options.

Motion list can be generated with 4-way direction or 8-way direction. This precision is sufficient enough since a pixel has maximum 8 neighbors. In both cases vehicle drive module is going to process these list and directed mobile robot toward its task. Heading of mobile robot is taken from coordinate of identification color pattern.

3 Demonstration

We run two different experiments in order to observe how our guidance control framework solves problems in these experiments (figure 5). First experiment is done for how simply framework can be use by programmers. Our framework can also be used from graphical user interface.

In code mode, basis of working sample is shown and explained in table 1. In the table standard includes are not written, only focus on framework's codes.

In graphical user interface (figure 5), the first step is to select connected capture device from camera list, then connect each mobile robot by selecting virtual communication port from COM port list, after this mobile robots' name, and battery information

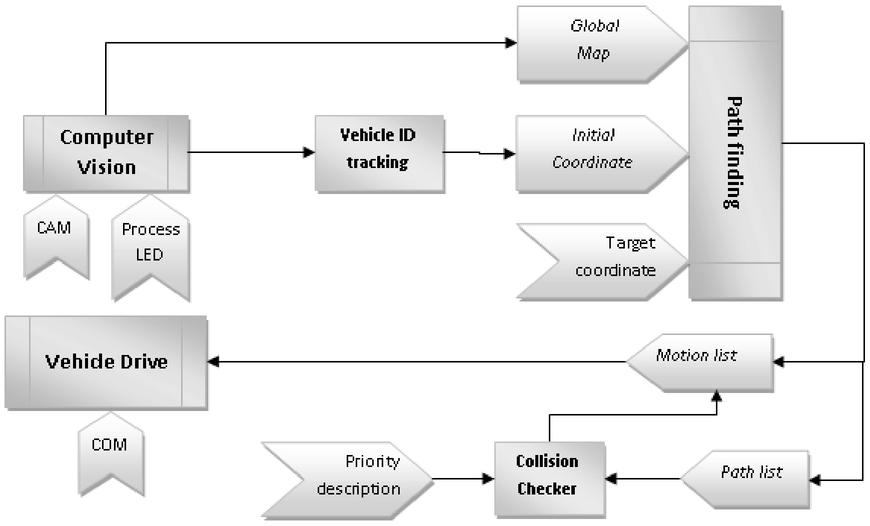


Fig. 4. Multi robot guidance control flow diagram

Table 1. Basic mobile robot guidance

Code sample	Explanation

`Pathfinder _pathfinder = new Pathfinder();`
`ILegoDriver legoDriver = null;`
`FilterInfoCollection localCameras = new`
`FilterInfoCollection(FilterCategory.VideoInputDevice);`
`camera = localCameras[0].MonikerString;`
`searchWindowLimit = 10;`
`int vehicleID = 1;`
`LegoDriver vehicleDriver = new LegoDriver("COM4");`
`VideoCaptureDevice videoSource = new`
`VideoCaptureDevice(camera);`
`videoSource.DesiredFrameSize = new Size(640, 480);`
`videoSource.DesiredFrameRate = 25;`
	Define first general parameters.

`vehicleDriver.processLED("on");`
`globalVision.searchPL(rgb(255,0,0));`
`globalVision.searchPattern(rgb(0,0,255),`
`globaVision.PLcoordinate.X,globalVisionPLcoordinate.Y);`
`List<Point> obstacle =`
`globalVision.searchObstacle(rgb(255,255,0));`
	Start the global vision module.

`Point iPoint = new Point(globalVision.PatternCoordinate.X,`
`globalVision.PatternCoordinate.Y);`
`Point tPoint = new Point(16,3);`
`_pathfinder.Squares[iPoint.X, iPoint.Y].ContentCode =`
`SquareContent.StartLocation(vehicleID);`
`_pathfinder.Squares[tPoint.X, tPoint.Y].ContentCode =`
`SquareContent.FinishLocation(vehicleID);`
	Assign start and end location for mobile robot.

`foreach(Point o in obstacle)`
`{`
`  _pathfinder.Squares[o.X, o.Y].ContentCode =`
`SquareContent.Obstacle;`
`}`
`Recalculate(vehicleID);`
`priority = _pathfinder.HighlightPath(vehicleID).Count;`
	Generate global map, then calculate shortest path and priority from path length.

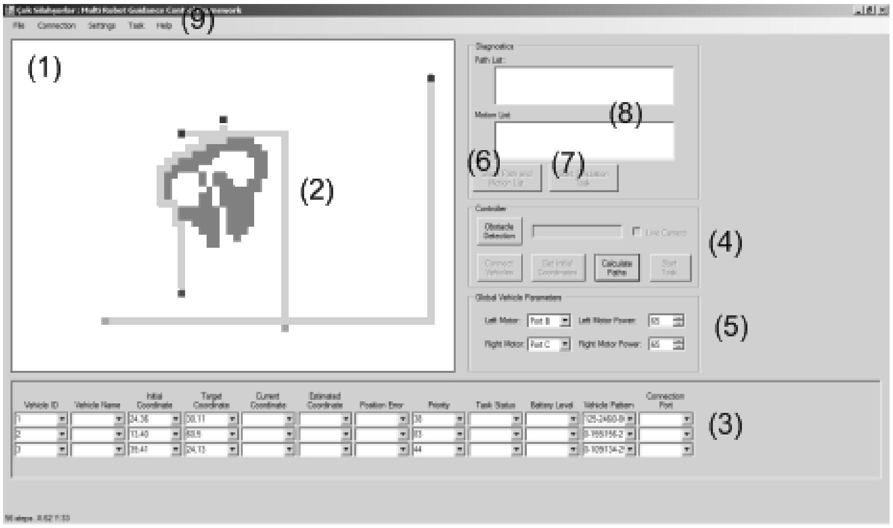
`if(vehicleDriver.Connect())`
`{`
`  List<Point> path = _pathfinder.HighlightPath(vehicleID);`
`  for (int i = 1; i < path.Count; i++)`
`  {`
`    Point movement;`
`    movement = Pathfinder.ExtractMovement(path[i - 1],`
`path[i]);`
`    vehicleDrive.run(movement);`
`  }`
`}`
	Connect mobile robot. Then generate motion list and run.

are taken if suitable ports are given. If user wants to set a virtual mobile robot, it can be manually added to the system. After the connection procedure, vehicle drive module start to scanning process LEDs, until computer vision module finishes getting process LEDs' and mobile robots' pattern coordinate. If the number of the mobile robots is more than ten, then we suggest selecting mobile robot when the process LED is active with the help of mouse cursor. This will be more fast, and easy. With the help of this process framework knows which communication port control which mobile robot, each mobile robot's name, battery level, identification pattern color, and current coordinate. The next step is to generate global map. This is computed by computer vision module, and obstacles are found by using connected components labeling. The next step is to assign task, for experiment it will be "going from A to B with shortest path". For this step path finding module computes shortest map according to global map and target coordinate for each mobile robot (table 2). When proper paths are found, priorities sorts which mobile robot is continue when paths are crossed. Then framework is ready to start the task for all system. When the task continues, mobile robots are tracked by computer vision module with the help of identification color pattern. The tracking can be very complex when the mobile robot population is large. User can change the sample time for tracking or it can also leave as real-time tracking.

Second experiment will demonstrate how multi robots can avoid randomly moving obstacles while running their own tasks. This will be our future work; it will be added in this paper later.

Table 2. Pseudo code for path finding

```
unvisitedList.Add(initialCoordinate)
while(unvisitedList.Count > 0)
  currentPoint = getLowestCost(unvisitedList)
  if(currentPoint == targetCoordinate)
    pathReady = true
  else
    visitedList.Add(currentPoint)
    neighbors = currentPoint.getAdjacentPointList
    foreach(point neighborPoint in neighbors)
      if(unvisitedList.isItem(neighborPoint) == false &&
        visitedList.isItem(neighborPoint) == false &&
        neighborPoint.content != content.Obstacle)
        unvisitedList.Add(neighborPoint)
        CalculateCost()
      end if
    end foreach
  end if
end while
```



Left Motor: Port B
Left Motor Power: 65
Right Motor: Port C
Right Motor Power: 65

Vehicle ID	Vehicle Name	Initial Coordinate	Target Coordinate	Current Coordinate	Estimated Coordinate	Position Error	Priority	Task Status	Battery Level	Vehicle Pattern	Connection Port
1		24.36	30.11				38			125-2460-0	
2		13.40	63.5				83			0-155156-2	
3		35.41	24.13				44			0-109134-2	

Fig. 5. Screenshot of framework's GUI with virtual mobile robots. (1) Live camera and path controller. (2) Mobile robots and their paths. (3) Mobile robot property controller grid. (4) Camera and vehicle connection, obstacle selection, vehicle recognition, path calculation and task starting buttons. (5) Global vehicle parameters (6) Path and motion list calculation control for selected mobile robot. (7) Simulation control of the defined system. (8) Path and motion list for selected mobile robot. (9) Standard controller menu for all parameters and help section.

4 Conclusion

In this paper, multi robot guidance control framework is introduced and explained how it is constructed and work with code and graphical user interface. Framework has design to be robust, reliable and programmer-friendly in the meaning of simplicity and easy to use. We successfully control our Lego NXT mobile robots for given tasks. We also can simulate virtual mobile robots for what-if scenarios by entering their coordinates, battery levels if we use battery level as a priority. With the help of this propriety mobile robot behaviors and success rates can be examine without setting up the real system in the field. And if the system fails from any problem, system does not cause any damage on any mobile robots. Framework will continue to be developed and it will present more options to the users. Especially vehicle drive module is going to drive mobile robots more smoothly not as see on figure 5 [3]. And computer vision module is going to recognize 2D barcodes for tracking and identification. This will be main consideration because in industrial applications color based tracking and identification cannot be accurate for mobile robots. For example in a factory, the environment which is controlled by a vision system cannot as clean as in an office floor or a laboratory. This will cause computer vision system is going to acquire disturbed color data, consequently identification process will become vulnerable. This type of vision systems is highly affected from light conditions, and needed professional camera solutions with precise calibration.

Acknowledgements

I wish to thank my advisor Prof. Dr. Erol Uyar for all his theoretical support. Dr. Levent Çetin and Ph. D. student Özgün Başer contributed greatly to the writing of this paper. Also my friend N. Engin Toklu provided valuable ideas for the developing of the framework. And finally Google Code, for hosting our framework [7].

References

1. Nabiyev, V.V.: Algoritmalar: Temelden Uygulamalara. Seçkin, Ankara (2007)
2. Amit's Game Programming Information, `http://www-cs-students.stanford.edu/~amitp/gameprog.html`
3. Ahmadzadeh, A., Jadbabaie, A., Kumar, V., Pappas, G.J.: Stable Multi-Particle Systems and Application in Multi-Vehicle Path Planning and Coverage. In: 46th IEEE Conference on Decision and Control, New Orleans (2007)
4. Dussutour, A., Deneubourg, J.L., Beshers, S., Fourcassie, V.: Priority rules govern the organization of traffic on foraging trails under crowding conditions in the leaf-cutting ant Atta colombica. Journal of Experimental Biology, 499–505 (2009)
5. Kolushev, F.A., Bogdanov, A.A.: Multi-agent optimal path planning for mobile robots in environment with obstacles. In: Bjorner, D., Broy, M., Zamulin, A.V. (eds.) PSI 1999. LNCS, vol. 1755, pp. 503–510. Springer, Heidelberg (2000)
6. Michael, G., Kay, M.G., Luo, C.L.: Camera Placement for Global Vision. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (1992)
7. "Çok Silahşorlar" (Multi Musketeers in Turkish):A framework for multi mobile robot guidance control, `http://code.google.com/p/coksilahsorlar`
8. Vlassis, N.: A Concise Introduction to Multi-Agent Systems and Distributed Artificial Intelligence. Morgan & Claypool (2007)
9. Stone, P.: Intelligent Autonomous Robotics: A Robot Soccer Case Study. Morgan & Claypool (2007)
10. Liu, J., Wu, J.: Multi-Agent Robotic Systems. CRC, Florida (2000)
11. Geramifard, A., Chubak, P., Bulitko, V.: Biased Cost Path finding. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE), California, pp. 112–114 (2006)
12. Tavakoli, Y., Javadi, H.H.S., Adabi, S.: A Cellular Automata Based Algorithm for Path Planning in Multi-Agent Systems with A Common Goal. IJCSNS International Journal of Computer Science and Network Security (2008)
13. Patin, F.: An Introduction To Digital Image Processing (2003)

Author Index