DISSERTATION

Knowledge-based Multi-Agent Architecture Applied in the Assembly Domain

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Em.O.Univ.Prof. Dipl.-Ing. Dr.techn. Gerfried Zeichen
E376
Institut für Automatisierungs- und Regelungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Dipl.-Ing. Munir Merdan
Mat.Nr.0126292
Albrechtsbergergasse 5/4, A-1120 Wien

Wien, im März 2009

MUNIR MERDAN

Abstract

Today's manufacturing systems are often too inflexible and not sufficiently adaptable to rapidly changing environments with unpredictable and abrupt fluctuation in product demands or manufacturing downtimes. Moreover, such systems suffer from the weak covering and exchange of the information and knowledge between enterprise levels.

Since the assembly participates with high percentage in the cost of manufacturing a product and since the automation rate in this domain is very low, this is where the most benefits can be gained by applying more flexible manufacturing paradigms. This dissertation intends to develop innovative, agile control architecture to face the current requirements imposed to the manufacturing enterprises in the assembly domain. Having their own problem-solving capabilities and ability to interact in order to reach an overall goal, the autonomous agents are considered as a promising approach to provide a suitable paradigm for designing intelligent manufacturing systems to enhance flexibility and agility. We propose a knowledge-intensive multi-agent architecture that enables ontology-based communication and cooperation among a set of autonomous and heterogeneous agents. An agent, the main core of our architecture, acts based on his knowledge, by sensing the manufacturing environment, triggering the reasoning process, which selects the proper actions to be executed and that will affect the manufacturing environment. Each agent has knowledge about his domain of application, about strategies, which can be used to achieve a specific goal, and knowledge about the (other) agents involved in the system. The agent is a representation of a manufacturing component that can be either a physical resource (numerical control machine, robot, pallet, etc.) or a logic entity (order, supply, etc.).

Having a shared ontology is critical for successful communication between agents, since such a shared ontology provides the common agreement and understanding about the concepts used. This offers the possibility for solving inter-operability problems. Therefore, the ontology will be shared among agents and will serve as the instrument to define the vocabulary used by the agents during their interactions, and to support "understanding" of the message content in the sense of its correct interpretation. In particular, the essential knowledge about the domain will be made available to the agents through an ontology.

Kurzfassung

Heutige Fertigungssysteme reagieren häufig zu inflexibel und unausreichend anpassungsfähig an die schnell ändernden Produktanforderungen mit unvorhersehbarer und plötzlicher Fluktuation sowie an die Produktionsstillstandzeiten. Außerdem leiden solche Systeme öfters unter einer schwachen Informations- und Wissensversorgung sowie unter schwachen Informationen- und Wissensaustausch zwischen verschiedenen Unternehmensbereichen.

Aufgrund dessen dass die Montage einen hohen Anteil an den Produktherstellungskosten hat und die Automatisierungsrate in diesem Arbeitsfeld sehr niedrig ist, führt die Anwendung flexiblerer Produktionsparadigma in dieser Domäne zu größten Nutzen. Das Ziel dieser Dissertation ist die Entwicklung einer innovativen, agilen Steuerungsarchitektur um die gegenwärtigen Anforderungen, die die Montage einem Produktionsunternehmen auferlegt, zu begegnen. Um ein gemeinsames Ziel zu erreichen, können autonome Agenten Probleme lösen und miteinander kommunizieren. Mit dieser Fähigkeit bieten sie einen viel versprechenden Ansatz zur Entwicklung passender Paradigma für das Design intelligenter Fertigungssysteme und zur Steigerung derer Flexibilität und Agilität. Unser Ansatz ist die Entwicklung einer wissensintensiven Multi-Agent Architektur, welche die ontologiebasierte Kommunikation und Kooperation zwischen autonomen und heterogenen Agenten ermöglicht. Ein Agent, das Kernstück unserer Architektur, agiert basierend auf seinem Wissen, indem er das Produktionsumfeld und -Bedingungen analysiert und den Schlussfolgerungs-Prozess auslöst. Dadurch wird die entsprechende Aktion ausgewählt, welche in ihrer Ausführung wiederum das Produktionsumfeld und bzw. Produktionsbedingungen beeinflusst. Jeder Agent verfügt über das Wissen über sein Anwendungsgebiet, über Strategien, die verwendet werden können um ein spezifisches Ziel zu erreichen, sowie über die im System beteiligten Agenten. Der Agent repräsentiert die Produktionskomponente, die entweder ein physikalisches Ressource (numerisches Steuerungsmaschine, Roboter, Ladeplatte, usw.) oder eine logische Einheit (Auftrag, Zulieferung, usw.) sein kann.

Eine mitbenutzte Ontologie ist entscheidend für eine erfolgreiche Kommunikation zwischen den Agenten, da sie ein gemeinsames Agreement und Verständigung über die verwendeten Konzepte liefert. Dies bietet die Möglichkeit die Interoperabilitätsproblemen zu lösen. Folglich, dient eine von Agenten gemeinsam benutzte Ontologie als Instrument dafür ein von Agenten, während ihrer Interaktionen verwendetes Vokabular zu definieren und „das Verstehen“ des Nachrichteninhalts im Sinne einer korrekten Interpretation zu unterstutzen. Insbesondere, das wesentliche Domänewissen wird den Agenten durch die Ontologie zugänglich gemacht.

Dedication

Dedicated to the memory of Professor Bernard Favre-Bulle

Acknowledgements

At the beginning, I would like to acknowledge the contributions of a large number of people, who in different ways have provided invaluable support for the completion of this thesis.

Most of all I would like to thank my first supervisor Professor Bernard Favre-Bulle, the professor, manager and samurai that embodied the best characteristics of all these professions. He was the motivator and spiritual mentor for my work, suffering me with the question “How is that better?”, but helping me also when I needed help the most. I wish I had more chance to learn from him. *Arigato gozaimasu Professor Favre.*

I gratefully acknowledge the help of Professor Gerfried Zeichen, who overtook in difficult circumstances the hard role of my first supervisor, giving me many insightful suggestions that enhance the overall quality of this thesis.

I have been more than privileged to have Professor Vladimír Mařík from the Czech Technical University as my co-supervisor who provided me valuable feedback concerning this thesis.

This dissertation would not have been possible without my colleague Gottfried Koppensteiner, who spent countless amounts of time and energy forcing the system to function properly. He shared with me the same excitement when the agents behaved appropriate and when I succeeded to drive snowboard 5 meters without a fall.

I would like to thank Dr. Alois Zoitl, Ingo Hegny, Christoph Ebm and other colleagues at the Automation and Control Institute for their encouragement, discussions and suggestions particularly in the field of Low-level control. Especially I would like to express my thanks to Wilfried Lepuschitz for many valuable comments on earlier drafts of this thesis.

I have to express my sincere gratitude to Dr. Pavel Vrba from Rockwell Automation, on whom I could always count on, for extremely qualified supervision, advices and critics on the style and content of this thesis. Thank you for the inspiration and for proving that the best multi-agent system is really – an implemented one.

Furthermore, I would like to thank Professor Stefan Biffl, Thomas Moser and Dr. Dindin Wahyudin from the Institute of Software Technology & Interactive Systems for working with me on distributed dynamic scheduling and the analysis of a range of workflow scheduling strategies based on multi-agent negotiation.

I am also grateful to Dr. Vedran Kordic and Dr. Edin Arnautovic for various scientific and technical debates as well as discussions about life in general we have had.

Significant contributions were provided by my students Lisa Vittori, Erhard List, Gabriel Weidenhausen, Stephan Auer and Benjamin Grössing, to whom I would like to express my heartfelt appreciation.

Finally, many warm thanks go to my family who provided constant and munificent encouragement. Thank you for supporting me during all these years and for pursuing me to finish this thesis as soon as possible. “Yes mother, one of my next serious projects will be marriage!”

This research was financed by the Austrian Science Fund (FWF), Projects No. FWFP19644-N13.

Content

Glossary of Symbols and Abbreviations

ACIN	: Automation and Control Institute.
ACL	: Agent Communication Language.
AI	: Artificial Intelligence.
AR	: Agenda Rerouting.
CA	: Contact Agent.
CAPP	: Computer-Aided Process Planning.
CDA	: Change-Direction-Algorithm.
CFP	: Call for Proposal.
CIM	: Computer Integrated Manufacturing.
CNP	: Contract Net Protocol.
CR	: Critical Ratio.
CR policy	: Complete Rerouting Policy.
CRT	: Critical Ratio + Transportation Time.
DCOM	: Distributed Component Object Model.
DF	: Directory Facilitator Agent.
DL	: Description Logic.
EDD	: Earliest Due Date.
ERP	: Enterprise Resource Planning.
FB	: Function Block.
FBA	: Function Block Adapters.
FCFS	: First Come, First Served.
FIPA	: Foundations for Intelligent Physical Agents.
GA	: Genetic Algorithm.
HB	: Heartbeat.
HLC	: High Level Control.
HLF	: High-Level-Failure.
JADE	: Java Agent Development Environment.
JESS	: Java Expert System Shell.
KASA	: Knowledge-based Multi-Agent System Architecture.
KQML	: Knowledge Query and Manipulation Language.
LLC	: Low Level Control.
LLF	: Low-Level-Failure.
MA	: Machine Agent.
MAS	: Multi-Agent System.
MAST	: Manufacturing Agent Simulation Tool.
NR	: New Jobs Rerouting.
OA	: Order Agent.
OWL	: Ontology Web Language.
PA	: Pallet Agent.
PLC	: Programmable Logic Controller.
PP	: Process planning.

RA	: Rockwell Automation.
RDF	: Resource Description Framework.
RDFS	: Resource Description Framework Schema.
RMA	: Remote Management Agent.
RS	: Right-shift Scheduling.
RT	: Real-time.
RT-UML	: Real-time Unified Modeling Language.
SA	: Supply Agent.
SIFB	: Service Interface Function Blocks.
SPA	: Shortest Path Algorithm.
SPT	: Shortest Processing + Transportation Time.
SP	: Shortest Processing Time.
TMS	: Test Management System.
XML	: eXtensible Markup Language.

1. Introduction

"Knowledge is of two kinds: we know a subject ourselves,
or we know where we can find information upon it."
Samuel Johnson

1.1 Background and Motivation

The manufacturing sector, faced with the growth in the variety of products and at the same time with a decreasing product life cycle, is forced by global competition to produce customized products in a short time at low price. It has to be capable to effectively react to sudden changes in customer demands, as well as to cope with unpredictable events such as failures and disruptions. However, the current manufacturing control systems, due to their centralized and hierarchical structure, respond weakly to the emerging challenges faced by new technological developments and market demands. Such systems don't have the ability to economically handle manufacturing of several products and variants with small lot sizes. They also have a limited capability for agile adaptation to unexpected internal and external disturbances. The lack of flexibility and adaptability of such systems in situations when particular resources become overloaded or unavailable directly influence the system effectiveness. This is particularly caused by a need to switch parts of the system off-line and recalculate previous plans and schedules according to the new system configurations, causing the loss of time and delays in production.

Moreover the applied control systems, usually consisted of heterogonous units which are using different type of data and data structures, are not capable to ensure the uninterrupted flow of information between and sometimes through the controlled levels. The applied methodologies in these systems are based on disconnected ordering, scheduling as well as execution processes and lack agility needed for enterprise-wide integration. The process planning is usually separated from scheduling as well as control activities and unnecessary breaks between implicated systems are created, even though the outputs and data from one application could be fluently used as inputs for another one. The committee on Visionary Manufacturing Challenges identifies the information and knowledge that covers and is exchanged between enterprise levels including the shop floor to be one of major challenges that need to be addressed when manufacturing companies progress from the current status to manufacturing in 2020 [Vis98].

In order to cope with the shortcomings mentioned above the new agile, more flexible and robust manufacturing paradigm, capable to handle ongoing changes in a manufacturing environment, changing rapidly system configuration and maintaining efficiency at the same time, is required. The paradigm has to be able to dynamically optimize a production schedule considering system capacity and time constraints. The concept should also support interoperability and reduce time and costs for getting high-quality and accurate knowledge through its information systems. Moreover, the concept has to be capable to integrate the product and manufacturing system life cycle giving them a unified and understandable form.

1.2 Approach

It has been generally acknowledged that the multi-agent system (MAS) approach offers a convenient way of modeling processes and systems that are distributed over space and time, making the control of the system decentralized, thereby reducing the complexity, increasing flexibility and robustness and enhancing adaptability to uncertainty and disturbances such as machine failures, frequent changes in the shop floor layout and the control system, etc [Bus04, She99, Vrb08]. MAS are composed of distributed heterogeneous units/agents, where each agent manages its own activities on the basis of its local state and the information received in messages from other agents or alternatively from human users (plant operators).

We propose a Knowledge-based Multi-Agent System Architecture (KASA) that enables ontology-based communication and cooperation among a set of autonomous and heterogeneous units/agents. An agent, the core component of our architecture, acts on the basis of his knowledge and by sensing the manufacturing environment. This triggers the reasoning process, by which the agent selects proper actions to be executed that subsequently influence the manufacturing environment. Having its own objectives, knowledge and skills, each agent has the capability to reason in order to take decisions about its activities. It is envisaged that adopting this control approach to manufacturing systems would improve agility as well as reusability and reduce the development costs.

The crucial element in the decision component is the rule-based system, which applies declarative knowledge, expressed in a set of rules to regulate the agent’s behavior. Agents communicate and negotiate with each other in order to perform the operations based on the available local information or to solve possible conflicts. The inter-agent communication capability provides the essential means for the collaboration of the agents. In order to ensure the correct understanding of the exchanged messages, all agents must have the same representation of the environment, or at least that part of the shared environment about which they are exchanging information with each other. Ontologies are of a vital importance for enabling knowledge interoperations between agents and at the same time a fluent flow for the different data from different entities.

In the proposed multi-agent system, each agent is an autonomous semantic entity responsible for the maintenance of the local data described in its knowledge base. Each agent is a representation of a manufacturing component that can be either a physical resource (numerical control machine, robot, pallet, etc.) or a logic entity (order, supply, etc.). In contrary to traditional manufacturing scheduling systems that are using centralized scheduling, the proposed agent-based manufacturing scheduling system supports distributed scheduling. Each agent can handle the schedule of its machine, operator, robot or station locally. In addition, this system optimizes machine utilization and provides a platform to enable the reconfiguration of manufacturing systems.

The major advantage of the proposed architecture is that using the ontology driven solution in combination with intelligent agents offers a direction towards solving the inter-operability problems within the manufacturing life cycle as well as between software applications, such as process planning, process modeling, scheduling, workflow and simulation. The proposed system automates the generation of process plans such that functions as for instance task selection, shortest route determination, etc. can be performed automatically through cooperation and coordination between autonomous agents without involving the central unit. Furthermore, the advantage of the presented concept is that in this knowledge-based system there is no need to define how a problem has to be solved (i.e. which detail actions have to be taken), instead the problem and the goals to be reached have to be described. The system then chooses on its own how to reach the goal.

Additional advantages of this approach are:

- Adaptation to disturbances - the agile response to unexpected manufacturing disturbances,
- Re-use of knowledge, know-how and components, simplification of the solution development,
- Addition of new components and knowledge, by avoiding re-design, re-programming and re-initialization of the other components,
- High degree of flexibility, modularity and reusability of hardware and software components,
- Generation of production tasks based on ontologies,
- Introduction of new products does not require significant time and efforts for programming and adjusting the system,
- Achieving the preconditions for easy assembly and disassembly of products.

1.3 Application

The proposed concept is applied in the assembly domain, since the assembly activities commonly make up on average 40% of product costs and 50% of production investments [Del96]. This is where the greatest competitive advantage can be gained by introducing more responsive manufacturing paradigms. Thus, the increased complexity of the manufacturing domain is especially crucial in the assembly area, which involves a lot of manual work, copes with shorter life cycles, many variants as well as smaller lot sizes of the product and is therefore responsible for a major part of the manufacturing costs [Bed91, Del96]. Furthermore, an assembly system has to be able to react on internal disturbances which occur during assembly and which endanger the operation of the system [Hei01]. There has been a clear recognition of the need for agile, knowledge intensive assembly systems that can easily absorb the required changes in product volumes, variety and manufacturing organization [Hei00].

As a basis of the developed architecture, the existing “Test bed for Distributed Holonic Control” at the Institute for Automation and Control[1], Vienna University of Technology, has been used and its structure and functions emulated. The Test bed architecture consists of an automatic storage system with a handling unit for the extraction of parts, a pallet transfer system with redundant paths, as well as a portal for the final assembly.

1.4 Objective

The primary scientific aim of this thesis is to investigate the applicability and effectiveness of a knowledge-based multi-agent architecture for managing distributed manufacturing systems. Furthermore, this thesis will also pursue research questions such as: how does the ontology affect the development of a distributed manufacturing system, which conditions should be fulfilled to support knowledge acquisition and sharing across its heterogeneous sources, can the use of ontologies really solve the interoperability problem in the manufacturing domain as well as which steps should be done and to ensure the easy “agentification” of manufacturing components?
In order to find answers to the questions above this thesis pursues the following research goals:

- Development of an innovative and agile architecture for distributed manufacturing control systems,
- Establishment of semantic interoperability among heterogeneous, enterprise levels,
- Design and employment of a persistent assembly ontology,
- Employment of knowledge-based techniques for improvement of the decision making process in intelligent agents,
- Creation of a facility that will enable the easier incorporation of agents into industry applications.

1.5 Methodology

A multi-agent system is a complex software system and its design and development is a complex process. This complexity is based on the distributed systems structure and reflects itself through the behavior adjustment of the individual agents to form the emergent system behavior. Several methodologies have been proposed for building MAS. The GAIA methodology has been presented by Wooldridge et al. [Woo00], MESSAGE by Caire et al. [Cai01] and Tropos by Giunchiglia et al. [Giu02]. Our approach for developing the KASA architecture is fundamentally based on a methodology for modeling multi-agent systems called Multi-agent Systems Engineering (MaSE) [DeL01]. The MaSE has been chosen since it provides a complete lifecycle methodology for developing MAS. However, since the mechanisms for the integration of ontologies and knowledge are not completely provided, we

[1] http://www.acin.tuwien.ac.at

developed these missing parts. The development of MAS is divided into the following three phases: analysis, design, and implementation (Figure 1.1).

The objective of the *Analysis* is understanding the system and its structure and the consideration of system requirements. This phase identifies all system goals, the manner how to reach these goals as well as relations and hierarchy between them. Besides, this phase divides the goals to related tasks and specifies the scenarios of their execution. Furthermore, the specific roles that can be responsible for the achievement of particular goals are also created and related tasks to be performed are assigned to those roles. These roles could be later transformed to particular agent classes. The phase finishes with the specification of the interaction model to support the goal organization that the system needs to accomplish.

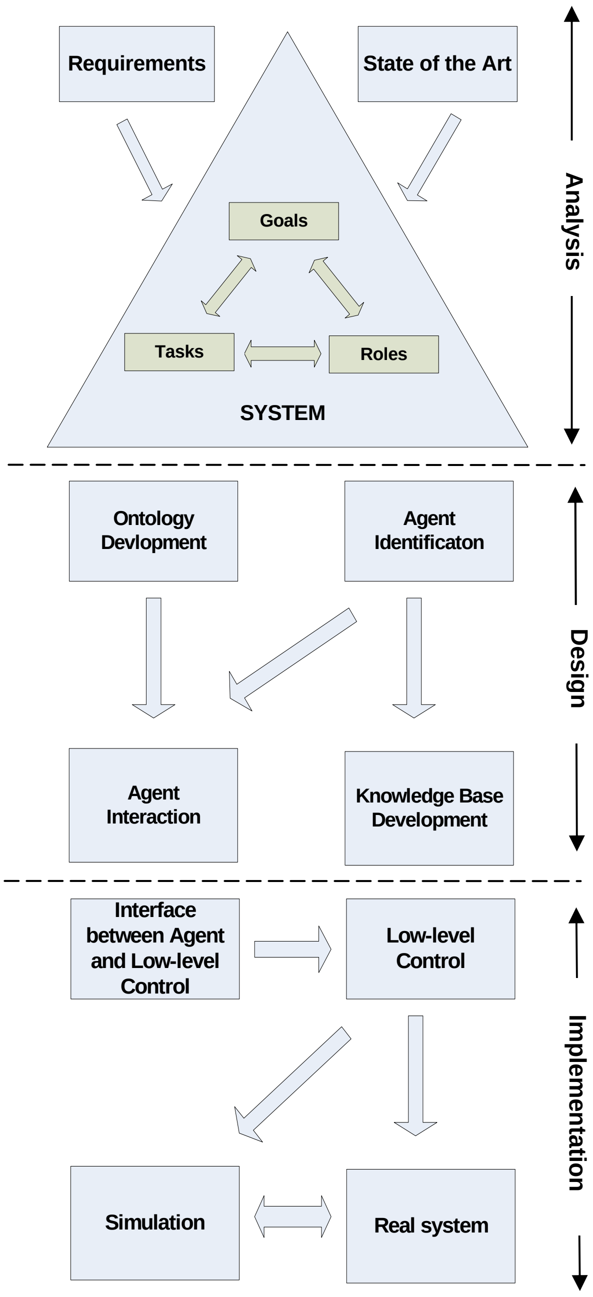


Figure 1.1: Methodology

The Design phase essentially consists of defining a solution for the system model, which was previously specified in the analysis activity. In the design process the following activities have to be performed:

- Identification of the number and type of the agents,
- Specification of the inter-agent relations and the required interaction protocols,
- Specification of ontology requirements (content, knowledge sources, potential users, usage scenarios, terms and potential relations between them, etc.),
- Development and refinement of the system ontology,
- Creation of the agent structure and development of the knowledge for each specific agent in the system,
- Analysis of the knowledge structure, preparation of validation scenarios and application of test scenarios to evaluate the software.

The system *Implementation* is started once the multi-agent system has been designed. Simulation is an indispensable tool for tuning and validating the agents' knowledge before they are applied to the real system. At the beginning of the simulation tests, the problem-solving and interaction abilities of the agents are investigated. After the instantiation of specified agents and setting related parameters, simple tasks are use to check for incompleteness in agents knowledge. The resolving of potential inconsistencies, redundancy and conflict in the agent's knowledge base enables to prevent possible. Furthermore, these tests are used for testing and validation of simulation environment.

In the second *"Simulation for verification"* phase, real assembly examples as well as routing trainings are used to test and validate the distributed system's functionality. The agility and performance of the architecture are studied simulating different scheduling scenarios as well as the failure of machines, particular agents or other system components e.g. conveyers. The results of the simulation, the main scope of this thesis, will be used as an input for the MAS improvement before its deployment in real-life control, which is our next future task, and real system results will be used for improving the simulation.

Due to the system complexity, the classical waterfall development model was used only for creating the basis system structure. We mostly used an iterative development process between phases, since the feedback from one phase had a lot of influence to the others. The feedback loop is used within as well as between the phases and repeated until the appropriate design is reached.

1.6 Thesis Outlines

The thesis is structured as follows.

Chapter 2 is divided in two subsections. On the one side, the first part of the chapter reviews the state of the art in the manufacturing area with a special focus on the manufacturing control, the basics of knowledge modeling with a special focus on ontologies as the crucial factor that captures semantics within the knowledge model as well as their application in manufacturing domain are reviewed. On the another side, the second part of the chapter gives the overview of the reported work related to production planning and scheduling considering basic preconditions and requirements needed for their integration. In this chapter, we also analyzed the recent research work done in the MAS domain specifying advantages of the MAS and presenting the main weaknesses and challenges that need to be solved.

Chapter 3 will describe the proposed multi-agent architecture and the role of each agent within this architecture. This chapter will also present the underlying concepts used to support the architecture comprising of a description of the system ontology, including the organization and vocabulary definition as well as the definition of the low-level control layer.

The basics of dynamic scheduling are introduced within the Chapter 4, including state of the art, requirements as well as the application scenarios. The scheduling algorithms and the rescheduling strategies for the reaction to disturbances are presented as well.

Chapter 5 presents the system failure tolerance concept. The implementation and experimental results, concerning stability and robustness of the system are reported in this chapter.

Chapter 6 explains the realization of the proposed approach and briefly describes the used tools as well as the way of their application.

Chapter 7 discusses the main results and the main achievements of this work. The possible directions for further research are presented.

2. State of the Art

"The farther backwards you can look,
the farther forward you are likely to see."
Sir Winston Churchill

2.1 Introduction to manufacturing systems

Over the history, manufacturing systems followed the evolution of the market trying to accommodate the customer demands. It started with Eli Whitney in 1798, when he announced his ability to produce, for that time, incredible 10000 muskets within 28 months. He pioneered with the standardization in production of muskets by designing the templates for each musket's part as well as by adding machines for the production of interchangeable parts. However, although he didn't reach his aim - completing his contract 10.5 years later - he set the basis preconditions for incoming mass production age [Boo05]. The basic principles of the mass production defined Frederick W. Taylor developing methods for the measure and design of machining methods. He used time and motion studies to analyze and split complex operations into a number of smaller and repetitive tasks in order to increase the production efficiency [Tay11]. This age really started when Henry Ford applied these principles and introduced a moving assembly line for producing the Ford model T automobile in 1913, reducing the production costs and at the same time increasing productivity and product quality [Bar04]. H. Ford could afford to say that "*a buyer could have any color of the car he wanted so long as it was black*", since he was able to offer the affordable product to the insatiable market at that time. This system fell apart in the 1970s due to bad attitude towards workers and its inflexibility to change according to customer demands looking for more quality and for customized products at a favorable price. Responding to these demands, the manufacturing domain invested more in automation and forced the "Just in Time" concept pioneered by Toyota producing only what is needed when it is needed, looking also for solutions in the emerging information technology based on computers called Computer Integrated Manufacturing (CIM) [Har73]. Nevertheless, these showed expected results in the 80s by evolving in Lean Manufacturing age which incorporated Japanese philosophy based on waste elimination, permanent improvement and efficiency maximization [Wom90]. However, this paradigm shows weaknesses when facing the continuous process of market changes characterized by the shorter life cycles, many variants as well as smaller lot sizes of the product. Furthermore, the global competition is forcing the manufacturing domain to produce customized products in a short time and at a low price. All these requirements imply and increase the complexity of the process control and planning, which are involving a huge amount of parameters that has to be considered and combined with characteristics of modern technology manufacturing systems that are already complex on their own. This increased complexity of current manufacturing systems together with dynamic conditions and permanent demands for flexible and fault-tolerant functionality makes their management and control very difficult and challenging. Manufacturing systems are expected to face all these requirements, achieving shorter lead-times and at the same time coping with external turbulences (dynamic market changes and technological challenges) as well as with internal disturbances (e.g. process failures or machine breakdowns). The new agile manufacturing paradigm is seen as a concept able to help manufacturing systems to enter the new production age [Kid94]. The concept of agile manufacturing offers the solutions that could enable the manufacturing systems to be more concurrent, flexible, adaptive and capable to rapidly reconfigure themselves. Gunasekaran defined agility "*…as the capability to survive and prosper in a competitive environment of continuous and unpredictable change by reacting quickly and effectively to changing markets, driven by customer-designed products and services….*" [Gun98]. The main strengths of the agile manufacturing paradigm are its concepts of a seamless information flow from the product design over production until its final delivery to the costumer as well as the new shop floor organization. The agile paradigm breaks also the company borders forcing the creation of "competent enterprises networks" (virtual organizations), where each company within the alliance offers specific services and products.

Summarizing this short overview it could be concluded that investments in innovation, standardization and automation, which lead to an improved flow of products, data, information and knowledge, have always brought the spearheads significant advantages in comparison with others.

2.2 Manufacturing Control

The factory control is defined "*…as the actuation of a manufacturing plant to make products, using the present and past observed state of the manufacturing plant, and demand from the market*". It is the fundamental system of a factory, because "*It coordinates the use of the factory's resources, giving the system its purpose and meaning*" [Bak98]. The manufacturing control can be divided into low-level and high-level control [Chr03]. The high-level part of the factory control is responsible for the coordination of the manufacturing resources and government of the production including the ERP (Enterprise Resource Planning) as well as the MES (Manufacturing Execution System) level. The low-level control is focused on the control of the individual manufacturing resources and their reliable function during the execution of operations organized on the high-level.

The manufacturing control architectures can be organized in a centralized, hierarchical, hybrid or heterarchical way as presented in Figure 2.1.

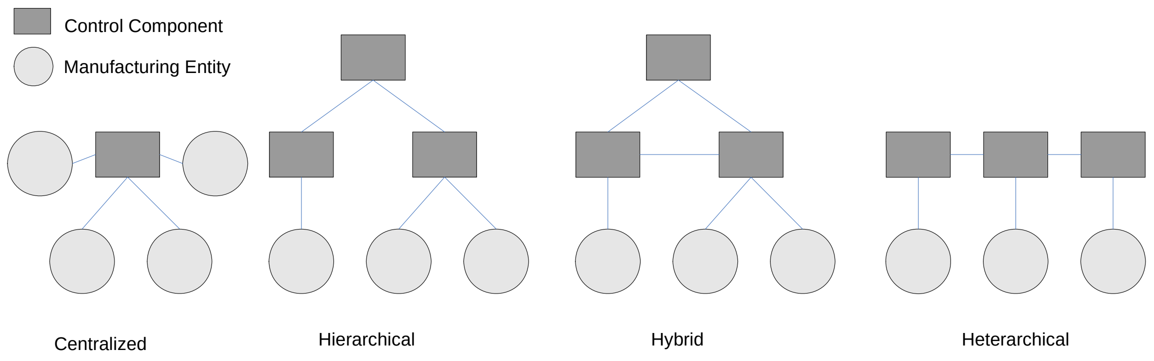


Figure 2.1: Control Structures [Lei04]

The centralized architecture is characterized by a central unit, which “sits” on a central database that provides a global view of the system, and does the control, planning and scheduling for a whole plant. The hierarchical architecture is built up in a pyramidal structure, where each node has its own purpose and tasks. In this architecture the flow of commands is typically top-down and the flow of information is bottom-up. The heterarchical architecture does not contain any supervisory controller or any other hierarchical element. The hybrid architecture is a combination of both hierarchical and heterarchical approach and allows direct communication between nodes in the control pyramid.

In current manufacturing systems, the hierarchical or centralized structures are the most commonly used system architectures. However, due to their rigid character and limited adaptation capabilities such systems respond weakly to frequently changing customer demands in terms of performing necessary changes in the manufacturing environment itself [Jon86, Par96, She99]. Present centralized and hierarchical control structures cannot dynamically manage the high degree of complexity – their modification is a very expensive and time consuming process. This is especially evident in the cases when certain resources become unavailable or additional resources are introduced to the system, since their rigid structure hinders a flexible redesign of the system. Additionally, the construction of a centralized system, due to large complexity and the necessity to centralize all logic for sensing, actuating and control into a single entity, usually requires a huge investment, long lead times, and in turn, results in generating a rigid control system [Col06]. The central controller, as it needs to have the accurate information about each unit in the system in order to make right decisions, can be seen as a single point of failure and its breakdown could stop the whole system [Kro99]. Scheduling, in centralized and hierarchical control structures, is established such that each level creates the scheduling for its subordinate levels having a weak feedback from lower levels and almost without any consultation and coordination with higher layers of neighboring units. Such an approach works well only if everything goes as expected; otherwise it could completely fail when unpredictable disturbances occur [Bus04].

The application of decentralized control architectures based on autonomous and co-operative units is considered as a promising approach for overcoming the weaknesses mentioned above [Fav04, Zei08]. Several emerging concepts like Holonic [Bru98], Multiagent [Jen98], Fractal [War93] and Bionic architectures [Oki93] are proposed as paradigms capable to handle the combinatorial complexity of manufacturing systems. These architectures, inspired by existing natural or social organization systems, are based on distributed entities (i.e., holons, agents, fractals or cells), which are using the self-organization principle to handle the system dynamics [Tha96].

2.2.1 Multi-agent Systems

The multi-agent approach has been widely recognized as enabling technology for designing and implementing the next-generation of distributed and intelligent manufacturing systems [She01, Bus04, Pec08]. Multi-agent systems can be defined as a network of autonomous, intelligent entities – agents – where each agent has individual goals and capabilities as well as individual problem-solving behaviors. Due to their lack of a global system objective and overview, agents have to cooperate and communicate with each other in order to achieve common aims, which are beyond the individual capabilities and knowledge possessed by each agent. This approach replaces a centralized database and control computer by a network of agents, each endowed with a local view of its environment and the ability and authority to respond locally to the environment.

The concept of an agent was invented in the domain of artificial intelligence (AI). There are several definitions of an agent, however the definition from Wooldridge integrates most of the agent characteristics "*an agent is an autonomous software entity that functioning continuously carries out a set of goal-oriented tasks on behalf of another entity, either human or software system. This software entity is able to perceive its environment through sensors and acts upon it through effectors, and in doing so, employs some knowledge or representation of the user's preferences*" [Woo99]. An agent:

- can make its own decisions and act autonomously in order to achieve its goals,
- can perceive its environment or interact with other agents in order to get the accurate environment representation,
- cooperates and collaborates with other agents, if it doesn't possess knowledge and expertise to accomplish its own goals;
- negotiates or competes with other agents in order to achieve better results,
- has the ability to acquire and to memorize new knowledge [Chr07].

Two different approaches for agent encapsulation in agent-based manufacturing systems are known: the functional decomposition approach and the physical decomposition approach

[She99]. In the functional decomposition approach, agents are used to encapsulate modules assigned to functions such as an order, task, etc. In the physical decomposition approach, agents are used to represent entities in the physical world, such as a robot, conveyor, pallet, etc. Based on the agent's behavior, the agents can be sorted into: reactive, deliberative and hybrid agents. Reactive agents are observing the environment and act to its changes in a reflexive way without maintaining any internal state or considering any historical information. These agents are usually used in architectures which have to satisfy real-time constraints. Contrary to reactive agents, deliberative agents behave like they are thinking, requiring accurate information about the status of the environment as well as historical information in order to make action plans and to predict the effects of actions. However, if the complexity of the problem significantly increases, their reliability and real-time reaction becomes questionable. A hybrid agent is a combination of the reactive and deliberative approach. Based on the mobility of an agent, it can be classified into a mobile or stationary agent.

Several different agent architecture types have been proposed in the literature: hierarchical, federate and autonomous [She07]. Hierarchical architectures correspond to the currently mostly used hierarchical organization of manufacturing companies. The agents are used to control particular components of the system, which are having hierarchical (master-slave) relationships, and by simply copying their functionality, the hierarchical structure of the agent organization is reproduced. The hierarchical type of organization could emerge by the usage of functional decomposition of some processes or entities that naturally have such a kind of structure (e.g. product order–work order–task, etc.). Within the federate based structures, the facilitator and mediator approaches are known as most dominant. The facilitator approach organizes agents in groups that communicate between each other through an interface called facilitator. The facilitators are responsible for the communication layer organization, messages translation, message routing and sometimes also for problem decomposition as well as scheduling and coordination between agents. Characteristic for this architecture is that there is no direct communication between the agents since the whole communication is transmitted over group facilitator. The mediator approach is based on mediator agents that can offer their services to all agents in the system. The mediators use brokering or recruiting mechanisms to find related agents, applying sometimes different techniques to learn from the agents' behavior. However, once the required agent has been found the further communication can continue with or without of interfering mediator agent which means that agents could communicate and cooperate directly.

The autonomous architectures are based on agents able to independently and individually handle their actions and states. Having their own behaviors, knowledge about other agents and environment representation, autonomous agents do not necessarily need the global overview to reach their goals. However, in the case of large amounts of agents the system could become too complex to be managed properly with highly unpredictable behavior.

a) Communication

Communication is one of the essential ways for agents to build and maintain the accurate representation of its environment. The exchange of the information enables agents to cooperate, negotiate and coordinate their actions. However, in order to "understand" each other (e.g. to interpret message correctly), agents need to use a common representation language and protocols as well as to share the semantic content of the represented knowledge [Fin94a]. In this sense it was necessary to develop an Agent Communication Language (ACL) that could be used as a way for exchanging information and for the embodiment of message content. The most widely used ACLs in multi-agent systems are the Knowledge Query and Manipulation Language (KQML) [Fin94b] and FIPA-ACL [Fou03]. Both languages adopted the speech act theory as a basis for agent communication [Aus62]. The speech act theory provides a clear way for expressing the meaning of the communicative actions carried out by the agents defining the types of messages and constraining the semantics.

The KQML has been introduced by the US DARPA's (Defense Advanced Research Projects Agency) Knowledge-Sharing Effort and is designed to support information and knowledge sharing among intelligent software agents offering a message format and a message-handling protocol. KQML is built on the open Internet standards and it is compatible with the TCP/IP, SMTP, and FTP protocols.

The Foundations for Intelligent Physical Agents (FIPA) organization, formed to produce software standards for heterogeneous and interacting agents and agent-based systems, presented FIPA-ACL which is similar to KQML. A comparison of both languages has been done by Labrou [Lab01]. A message written in FIPA-ACL consists of the following elements: performative, sender, receiver, content, language, encoding, ontology, protocol, conversation-id, etc. The fundamental elements are language, performative (e.g. INFORM, REQUEST, CFP, PROPOSE, AGREE, FAILURE, SUBSCRIBE, etc.), content and ontology. In this dissertation, we used FIPA-ACL for communication, RDF as content language and XML as an encoding syntax. Nevertheless, the content of an ACL message received by an agent can be understood only if it shares a common ontology with the agent that sent the message [Gun05].

b) MAS Advantages

Making the control of the system decentralized, intelligent agents offer a convenient way of modeling processes and systems that are distributed over space and time, thereby reducing the complexity, increasing flexibility and enhancing fault tolerance [Jen03]. It has been suggested that the multi-agent system approach is especially adequate for the solution of problems with a dynamic and uncertain nature [Ald04]. Nevertheless, the agent-oriented concepts are as well suited for developing and extending complex, distributed systems by providing the most natural means of representing the distinct individuals and organizations, suitable abstractions, the ability to wrap legacy systems and flexibility for organizational structure changes [Jen01]. Generally, the introduction of agent-based techniques can bring the following benefits [Syc98, Sto00]:

- Robustness – sharing the control capabilities among the different agents, the system is able to tolerate when one or more agents fail.
- Scalability – it is easier to add new agents possessing new capabilities to the distributed system, than it is to extend the rigid structure of a centralized system by adding new functionality.
- Reusability – the development and maintenance of a modular system is easier than of a monolithic one. Agent classes with specific behavior to control particular manufacturing equipment developed for one application can be further reused when developing another control application.
- Parallelism – distributing the system and applying multiple agents can enhance the overall system performance and provide time savings by deploying parallel computation.

Respecting its advantages, a lot of research has been done and reported with regard of the Considerable research results have been achieved in the application of MAS to a wide range of manufacturing tasks including low-level, shop floor control, process planning and scheduling, information and project management as well as modeling of logistics systems, supply chains and virtual organizations.

In the next few subsections we will elaborate the relevant and recent work related to the manufacturing control area concerned with the high-level and low-level control. Furthermore, weaknesses and challenges of current applications will be identified and some solutions suggested.

2.2.2 High-level Control

One of the earliest MAS applications was the YAMS (Yet Another Manufacturing System) system [Par87]. YAMS modeled a manufacturing enterprise as a hierarchical system made of the following entities: plants, flexible manufacturing systems, workcells and, on the bottom, workstations. In contrast to traditional hierarchical systems, each entity (here represented as an agent) is able to not only negotiate with its parent and children, but also with its siblings as well. Each agent has a collection of plans representing its capabilities and uses the Contract Net Protocol (CNP) for inter-agent negotiation [Smi80].

Peng et al. presented a multi-agent consortium CIIMPLEX established for supporting the intelligent integration of manufacturing planning and execution, especially in managing the exceptions in business scenarios [Pen99]. Besides the common service agents, the gateway agent is used to provide an interface between the agent world and the application world making connections between the transport mechanisms and converting messages between the two different formats. The consortium introduces also several specialized agents such as data-mining/parameter-estimation agents needed to prepare the aggregated information about low-level activities for higher level analyses by other agents, event monitoring agents, which detect abnormal events, the analysis agents that evaluate disturbances of to the current planned schedule and recommend appropriate actions as well as the scenario coordination agents which assist human decision making for specific business scenarios. The proposed multi-agent system is demonstrated through an example integration scenario involving real planning and execution software systems.

Oliveira [Oli94] implemented a heterarchical multi-agent architecture for an assembly robotics cell. Intelligent sub-systems (agents) representing the specific functionalities are: task planner, execution planner, object recognition systems, data base and world descriptor that coordinates other agents' activities in order to minimize the occurrence of conflicts.

The needs-driven AARIA (Autonomous Agents for Rock Island Arsenal) agent system, developed for an Army manufacturing facility, encapsulates agents as representation of parts, resources and unit processes [Par98]. Each agent interoperates with other agents in and outside its own factory and uses the market-driven, inter-agent coordination approach for schedule optimization. The agent infrastructure supports broadcast and multicast communication, subject based addressing for inter-agent, location-independent communication as well as dynamic mechanisms for agent creation, migration, duplication and termination. The *AARIA* architecture is implemented on top of the *Cybele agent* infrastructure. The improved system performance related to the systems agility and equipment utilization has been reported.

MetaMorph I is a multi-agent architecture for intelligent manufacturing built as a federated organization [Mat96]. There are two main types of agents: resource agents (used to represent manufacturing devices and operations) and mediator agents. The mediator agents have the role of system coordinators and use brokering and recruiting communication mechanisms to find related agents for establishing collaborative subsystems. To support coordination, the architecture employs learning mechanisms for learning from the simulated future as well as for learning from the history. With the objective to integrate design, planning, scheduling, execution, etc. the MetaMorph II extends the previous MetaMorph I architecture and uses the hybrid approach for organizing and integrating the subsystems at the highest level through special mediators [She00a]. Each subsystem, which can be an agent-based system (manufacturing scheduling, management, etc.), can represent a complex system that is composed of other subsystems. However, agents in particular subsystem have the ability to communicate directly with other subsystems or with the agents in other subsystems at the same or different levels. The simulated architecture consists of four mediators: the Enterprise Mediator which is having administration role registering all other mediators, the Design Mediator which is used to integrate a feature-based intelligent design system, the Resource Mediator that provides the high-level coordination for a manufacturing shop floor, and the Marketing Mediator for the integration of the customer services into the system. The applied architecture features a reduced communication, easy integration of legacy systems and maintenance as well as considerable flexibility and scalability of the system.

Bussmann and Schild reported the development of a flexible transportation system and the associated agent-based control within the frame of the Production 2000+ project [Bus01]. In their architecture, a specific agent is associated with each workpiece, each machine, and each shifting table. The overall goal of the system is to continuously optimize the throughput while machine agents manage buffer sizes, workpiece agents manage the processing state of a specific workpiece and shifting table agents try to optimize the routing. The dynamic task allocation is organized as an auction where workpiece agents sell their current tasks, whereas machine agents bid for tasks taking into account the machine's current workload as well as the workpieces leaving the machine. The system has been evaluated in a series of simulations based on real manufacturing data (product types, processing times, disturbance characteristics, etc.). The simulations have shown that the agent-based control is extremely robust against disturbances of machines as well as failures of control units achieving a performance near to the theoretical optimum. In addition, the control system has been installed in the DaimlerChrysler plant in Stuttgart validating the results of the simulations under real manufacturing conditions. However this installment has been removed after some testing time because of two reasons. First, the used control hardware for the agent platform were standard PCs and therefore not suitable for an industrial environment. Second and more severe problem was the inability of the maintenance personnel to easily determine if the plant is working correctly or not because of the dynamically changing resource allocation as a result of agent negotiation.

Rockwell Automation, Inc. has investigated and implemented agent-based solutions for a number of industrial applications. One of the very first Rockwell Automation (RA) industrial agent projects was to increase the machine utilization of a steel rod bar mill at BHP Billiton, Melbourne in the mid nineties. The agent-based control was applied for the dynamic selection and configuration of a subset of working cooling boxes and rolling stands to satisfy the recipe requirements. In order to avoid the enclosure of the broken units in a subset, each cooling box and rolling stand was represented by an agent with the capability to assess its own health and bid on its part of the operation. Because of safety concerns and the anxiety of possible damage to the equipment, the agent-based control system did not directly control the bar mill but instead recommended a configuration to the human operator [Mar05a].

Another successful deployment of a multi-agent system by Rockwell Automation was the distributed control of a ship equipment applied to reduce the manning as well as to improve readiness and survivability of US Navy shipboard systems [Mat04]. The implemented system architecture is based on three hierarchical levels: the Ship-level is responsible for ship-wide resource allocation and communication with the ship crew; the Process-level optimizes the performance of the automation components and ensures the availability of services; and the Machine-level, as the lowest one in the hierarchy, focuses on control, diagnostics and reconfiguration of the shipboard equipment. The architecture of an agent consists of four main modules: planner, device model, execution control, and diagnostic. A simulation environment as well as physical, table-top demonstrator was built in the Rockwell Automation's research laboratories in Cleveland to allow testing the agent system functionality in a set of scenarios that mimic the transactions of the real shipboard system. Subsequently, the agent-based control system was successfully deployed on a physical, scaled-down model of the US Navy ship with agents running on standard PLCs of Logix family. The applied architecture provides a simple manner to establish system functions and dynamically emerging relationships among the system components without pre-programming the relationships. Further achieved advantages of the presented solution are: the system scalability, the ability to reconfigure complex systems in a fast way (within seconds) and the ability to continue in operation with partially damaged equipment [Tic06].

A simulation test-bed for the evaluation of agent-based distributed shop floor scheduling and control system is presented by Brennan and O [Bre00]. The architecture consists of four basic agents types, related to the basic holons used in the PROSA architecture [Bru98]: job agent, station agent, machine agent and mediator agent. The job agent is responsible for initiating the auction-based bidding process to find the resources for the job's operations, and monitoring the job's production progress. The station agents are responsible for the task assignment to specific machines, to monitor the production progress of the machines and to response to the job agent's bidding request. The machine agent, depending on the functional limitations of related low-level controllers, controls a particular machine. It can perform either simple operations or participate in a bidding process when bearing similar responsibilities as a station if the related related controller has the information processing and communication capabilities. The mediator agent registers the manufacturing resources and responds to the job agent's queries regarding which resource in the system can perform a particular type of operation. The architecture is evaluated on a test bed implemented in the Arena discrete event simulation package.

The multi-agent-system for production control of semiconductor wafer fabrication facilities called FABMAS is reported in [Mon03]. Due to the physical decomposition of the shop-floor into work areas and on the next level into machine groups that contain parallel machines, the agent architecture based on PROSA [Bru98] is organized in three layers using the following decision-making agent structure: single production system agents, multiple work area agents and multiple machine group agents. There are also additional agents used for modeling jobs and batches – they are not considered as part of a hierarchy and so they do not have any communication restrictions. The discrete event simulator AutoSched AP is used to simulate the behavior of the wafer fabs shop floor which consists of over 200 machines that are organized in about 80 different machine groups forming five work areas. The blackboard mechanism is used to exchange the sensor and actuator signals between the simulation and the FABMAS agent system.

Lastra et al. developed and applied the concept of intelligent mechatronic devices (actors) to the assembly domain [Mar05]. This concept – Actor-based Assembly Systems (ABAS) – is built on autonomous mechatronic devices (such as robot arms) that deploy auction- and negotiation-based multi-agent control in order to collaborate towards a common goal. This goal is a composition of simpler activities (assembly operations) which are the individual goals of the particular actors. A 3D visualization and simulation environment for the emulation of actors consists of two software tools: ABAS WorkBench and the ABAS Viewer. ABAS WorkBench, used for both modeling and emulation of actors, provides the designers with the ability to produce actor prototypes and experiment with them prior to the real implementation. The main goal of the ABAS Viewer is to monitor the performance of ABAS systems, serving as a runtime platform where actor societies can be deployed and visualized in a 3D environment.

Candido and Barata presented a multi-agent application developed to control a shop floor system [Can07]. The approach that follows some guidelines from the CoBASA (Coalition Based Approach for Shop Floor Agility) reference architecture [Bar05] is applied to the NovaFlex manufacturing system at the Intelligent Robotic Centre in UNINOVA, Portugal. The system is composed of two assembly robots, an automatic warehouse and a transport system that connects all these modules. Each shop floor component is abstracted as a Manufacturing Resource Agent (MRA) providing basic skills. More MRA agents can be aggregated to form a coalition providing complex skills. The Coalition Leader Agent gathers coalition members' basic skills to find out what complex skills (composition of elementary skills) could be supported by the coalition. The architecture provides also the Broker Agent, which gathers information from the environment and supervises/supports the process of creating the coalition. In order to maintain MRAs independently from a particular controller, a software Agent-Machine Interface wrapper is used to execute the skills exposed by the MRA and in order to abstract the lower level interaction with the hardware. An ontology is used to ensure an accurate information exchange, as well as to define the domain and relations between entities.

The successful deployment of the Manufacturing Agent Simulation Tool – MAST – in the manufacturing testbed at the Automation and Control Institute (ACIN), Vienna University of Technology (Figure 2.2) has been reported by Vrba et al. [Vrb08]. The MAST system is used to control the physical palette transfer system of the testbed consisting of 14 conveyor belts, 12 diverters, 4 docking stations and a number of sensors and stoppers. Three types of agents are used for representing the physical components (diverter, conveyor and docking station), whereupon each component has its own agent instance associated. The conveyor system is used to transport the palettes carrying raw materials or products between the docking stations, where the palettes are held until a particular manufacturing process is finished. The palettes are routed by the diverters that use reachability knowledge to send the palette to its required docking station at the lowest costs. The pallet agent selects the next docking station by searching the Directory Facilitator (yellow-pages services provided with the agent platform – JADE in this case [JAD08]) for a list of registered docking station agents. A failure scenario was tested – after a conveyor had failed (by turning off its drive), the conveyor agent notified its neighboring agents (representing for instance diverters or docking stations) that the conveyor is not available any more.

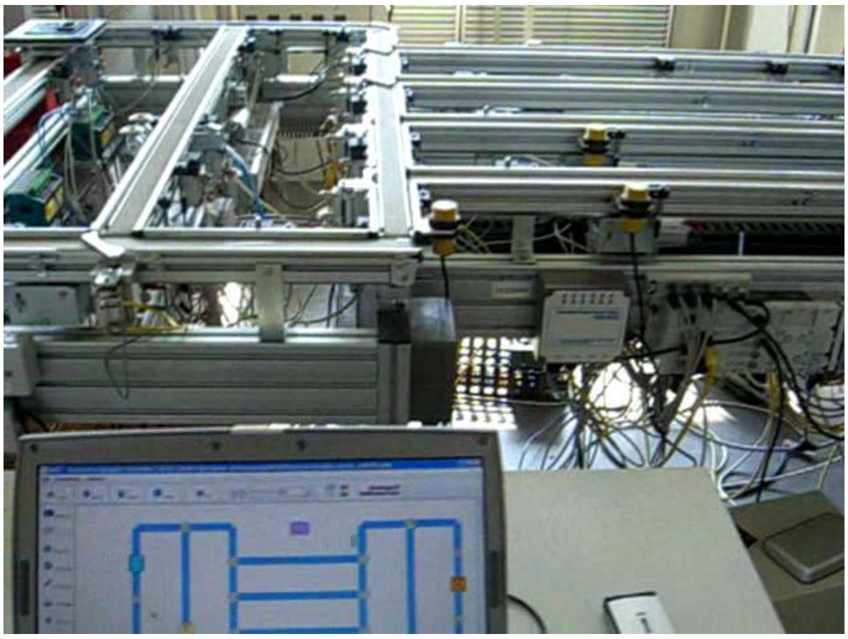


Figure 2.2: MAST application on the palette transfer system of the Odo Struger laboratory (ACIN, Vienna University of Technology)

Subsequently, the agents started to look for alternative routes for pallets to avoid the broken area. To increase the level of robustness of the deployed control system against non-determinism and disturbances of the real world, an embodiment of diagnostics to the agent as well as to the low-level control parts has been suggested.

2.2.3 Low-level Control

Applied in the manufacturing environment, the control system has to pay attention to the real world conditions and run under real-time constraints. The outcome of the performed action is not only based on the related reasoning but also on the time within which the action is executed. This is particularly important for subjacent embedded systems due to their responsibilities for directly dealing with the environment and synchronizing of their operations. These systems have to be able to observe the variation of the environment in order to immediately execute a set of certain actions or to send the accurate information to the high level agent-based control system. The linkage between the physical pieces of equipment controlled by the agent and the specific agent itself, is seen as a key characteristic of the application of agent technology for industrial control purposes [Vrb08].

One suggested approach is coupling the control and automation functions, whose response times range from 10 to 100 ms, in to the Low Level Control (LLC) layer as suggested by Christensen in the HMS architecture [Chr03]. The LLC is a user defined software application based on the IEC 61131-3 [IEC31] or IEC 61499 [IEC99] standards, which governs the actions of the subjacent physical system. The LLC is responsible for acquiring system information via sensors, executing particular control actions and sending signals back to actuators to carry out an action in the physical world. The LLC layer interacts with the HLC (High Level Control) layer. The HLC delegates particular tasks to the LLC and thus indirectly controls their execution. On the other hand, the LLC informs the HLC on the fulfillment or failure of the delegated tasks. For example, the Index station agent residing in the HLC instructs the LLC to hold a particular pallet when it arrives or requests to release it after the robot finishes a particular operation. On the other side, the LLC fixes the pallet when it arrives and then informs the agent about the arrival. The LLC can also diagnose some kind of failure (e.g. climb stock) and inform the agent about it. Built on top of the mechatronic components, the LLC layer runs on real-time embedded control equipment which includes the interfaces to the mechatronic components and the control application. In the case of industrial automation such equipment is typically a Programmable Logic Controller (PLC). Based on the currently most used IEC 61131-3 standard [Joh01] for programming PLCs, several concepts have been applied to establish these interfaces: Function Block Adapters (FBA), COM/DCOM technologies, black board or data table sharing. The first concept uses FBAs as interface between Real-time Unified Modeling Language (RT-UML) capsules and IEC 61131-3 function blocks. The behavior of an FBA is expressed by the special FBA-Language which defines operations which are called when signals arrive from a port or from the function block [Hev01].

The design and realization of an agent-oriented control system based on DCOM (Distributed Component Object Model) in simulation and under real shop floor conditions was done by DaimlerChrysler AG within the frame of the Production 2000+ project mentioned earlier in this thesis. A specific COM module, which encapsulates the algorithm for any agent type, is linked at runtime to a DCOM-based communication framework enabling the interactions with PLC programs controlling operations of the physical hardware [Sch00]. The discrete-event simulation model of a manufacturing system developed in C++ that utilizes COM/DCOM technology is presented by Brennan and O [Bre00]. The agents, encapsulated in COM objects, represent specific entities in the production system that is simulated in the Arena framework.

Agents, distributed over a network of computers communicate with Arena using the remote procedure call [Bre00]. The Agent-Machine Interface is the agent wrapper developed using DCOM and used in the NovaFlex environment to keep the agent behavior independent from the implementation details of the specific low-level control actions performed on PLC controllers. PLC manufacturer's proprietary Java interface libraries are used for reading and writing tags the Beckhoff PLC over the Ethernet to interact with IEC 61131-3 control programs [Can07].

Concerning MAST, the data-table concept is used for the implementation of the PLC-based control interface enabling the agents to interact with the physical system. The I/O values, transferred via the DeviceNet industrial communication network and stored in ControlLogix PLC tags, are directly accessed by MAST agents to allow monitoring and controlling of particular physical components [Vrb08]. Testing this architecture on a physical palette transfer system installed at the Automation and Control Institute (ACIN) at the Vienna University of Technology acknowledged the need for the underlying LLC layer for handling the low-level real-time activities. The agents (implemented in Java/C++), directly handling I/Os without presence of a LLC, revealed insufficient capabilities for meeting the hard real-time constraints. The recommended solution is to let the agents making global decisions and leave the low-level control subsystem to transform these decisions into the corresponding actuation in the real world.

A similar approach was presented by Rockwell Automation within its Shipboard Automation project. The Autonomous Cooperative System (ACS) architecture (implemented in C++) was introduced to provide an agent runtime environment for the standard Rockwell Automation ControlLogix^{TM} and FlexLogix^{TM} controllers. The firmware of the controllers was modified to enable hosting of 1-to-n of intelligent agents directly inside the controller. The ACS infrastructure enables also the distribution of agents over multiple controllers where they run in parallel and interact with the low-level control tasks (written in IEC 61131-3 ladder logic) by accessing the data-table of the controller [Mat04].

The control architectures for real-time and distributed manufacturing systems set a focus to a number of functionalities such as: autonomy, flexibility, reliability, fault-tolerance, interoperability, re-configurability, etc. The applied conventional systems, which are predominantly programmed according to IEC 61131-3, suffer often from a lack of these requirements [Wan98]. The system’s flexibility and disturbance handling capabilities of IEC 61131-3 are limited due to its centralized nature and difficulty to manage changes dynamically [Bre07]. Furthermore, the cyclically scan-based execution model of its programs is sensitive to the order in which functional elements are placed in the program and specific complex synchronization problems in distributed environments could occur. Especially timing software switches as well as synchronizing internal states are complicated [Zoi06a]. Moreover, the insufficient resource replacements and only predesigned reconfiguration scenarios embedded in the application hinder its wider application in distributed environments.

The new standard – IEC 61499 [IEC04] – introduces event-driven function blocks (FB), offering a framework for the integration of run-time control, diagnosis applications as well as simulation for distributed automation [Vya02], and provides basic support for reconfiguration at runtime [Zoi06b]. Although many researchers have already been investigating different aspects of IEC 61499, the absence of tools and products that are compliant with this approach is evident [Thr05 and Zoi07]. An important aspect of this research is the establishment of a run-time communication interface allowing the transfer of information from the real-time control subsystem to the agent-based high-level control layer and vice versa. Two recent papers have reported the deployment of such an interface. Lopez and Lastra used IEC 61499 to build a low-level FB application, which controls the environment by accessing the sensors and actuators, and which enables the agent level to perceive the environment [Lop07]. Special interfaces were developed that allow the communication interaction between components as well as internal communication between the low-level FB control and the high-level agent layer. A similar approach was reported by Hegny et al. [Heg08] that proposed the ontological representation of the communication channel and the message type to the agent level. This representation gives the agents the ability to reason about the purpose and meaning of the messages exchanged with LLC (e.g., sensor status information, device diagnostics, etc.).

2.2.4 Weaknesses and Challenges of Agent-based Control Systems

Although confirmed as a promising approach and deployed in a number of different applications over the past few years, the widespread adoption of agent-based concepts by industry and governments is still missing. The following weaknesses could be pointed out as main reasons for that:

1) Lack of awareness about the potentials of agent technology in industry as well as the absence of publicity of successful industrial projects [Pec05].
2) Lack of standards and methodologies that could simplify the integration of this technology in the manufacturing domain.
3) Missing trust in the idea of delegating tasks to autonomous agents [Syc98], especially considering emergent behavior of the overall control system (an aggregation of “small” behaviors of particular agents) that can hardly be predicted at the design time [Par97].
4) A “pioneer” risk, which accompanies every new technology that has not been proven in large scale industrial applications [Pec08].
5) Lack of design and development tools mature enough for industrial deployment [Pec08].
6) Paradigm misunderstanding due to the lack of real industrial applications [Can07].
7) Current applications are custom-developed for every single implementation, which means that costs can be spread neither over multiple customers nor over time [Val04].

8) Concerns regarding the stability, scalability and survivability, especially in unpredictable environments of attacks and system failures [Hel04].
9) Increased complexity of the software structure; due to the fact that in distributed environments each entity has to have the accurate status of the environment in its application domain, corresponding mechanisms have to be introduced to ensure the reliability of the environment representation and the functionality of the system.

Additionally to the weaknesses mentioned above, we also present research challenges that have to be fulfilled and considered to make a progress in the development and adoption of agent technologies:

1) Achieving interoperability in the heterogeneous manufacturing environment is one of the most important challenges for distributed manufacturing control systems. Interoperability is considered as the seamless flow and share of data, information and knowledge among heterogeneous entities as well as among subordinated information systems. Semantic systems and ontologies as presented in Sect. 2.2.5 are expected to provide effective means for fulfilling these ideas.
2) A further challenge is the linkage of the agent-based system to real-time information and its integration with the FB-technology [She06]. The inability to maintain an accurate representation of the environment in which an agent operates hinders effective task planning and execution.
3) Direct transformation of received raw data into knowledge compliant with formally defined semantics and effective reasoning on this data is also considered as a big challenge [Vis98]. According to Morel et al. "*a form of technical intelligence that goes beyond simple data through information to knowledge is required. Such technology embedded into manufacturing system components and within the products, will play an essential role to reach agility in manufacturing over flexibility and reactivity*" [Mor05].
4) "*To enable two devices with no previous knowledge on each other's type, conceived using different paradigms and interaction models but still with complementary skill sets, to interact autonomously*" [Las06].
5) The support of principles such as generality, reusability and long-term usage is seen as an important future challenge for software specifications [Val04].
6) Introduction of tools, techniques and methodologies which ensure easier and more abstract ways of agent system development, modification and management will lead to a higher rate of acceptance as well as "understanding" of the concept.
7) Assurance of security and trust in agents is a significant aspect to be considered in future solutions [Pec08]. Being applied in the manufacturing domain consisting of heterogonous entities and/or organizations, agent technology has to provide confidence that performed actions are done safely and effectively.

Summarizing the challenges it can be pointed out that besides the requirement for more efficient techniques and tools, the information and the ways of its handling are placed in the focus of future research. Completely new approaches where the information will be not treated as a "bunch" of strings and numbers but will be embedded with semantic providing the basis for its better "understanding" are required. Almost the same weaknesses detected by agent systems could be specified by the introduction of the IEC 61499 standard in the industry. However, the open problems as well as key research topics have been listed in depth by [Bre07, Thr05, and Zoi07].

2.2.5 Semantic Systems

Being applied in distributed and heterogeneous environment and having only its partial representation the agents have to communicate with each other in order to coordinate their activities. Ontologies have been developed and investigated for quite a while in artificial intelligence and natural language processing to facilitate knowledge sharing and reuse [Kul05]. They are of vital importance for enabling knowledge interoperations between agents and, at the same time, a fluent flow of different data from different entities. Ontologies allow the explicit specification of a domain of discourse, increasing the level of specification of knowledge by incorporating semantics into the data, and promote its exchange in an explicitly understandable form.

An ontology is defined as an explicit specification of conceptualization [Gru93], where conceptualization means the shared view of environment representation. From the viewpoint of inter-agent interactions, the explicitly defined and commonly accepted ontology is an indispensable tool for ensuring interoperability between agents in the sense of providing a formally defined specification of the meaning of those terms which are used during the inter-agent communication. Ontologies can also capture actions and events in a uniform and processable way so that they can be recorded in time and further analyzed.

An ontology includes classes, slots, relationships between classes, constrains about these classes and individuals. Classes represent a specific concept within the domain of interest. Slots represent properties of the classes. Constraints are used to define allowable values and connections within an ontology (Figure 2.3). Individuals are specific objects that instantiate the class and inherit its properties and relationships. If used for the description of complex domains, an ontology requires choice of an expressive language able to represent a particular domain. UML [UML], any object oriented language, or any other type of representation which can define objects, properties and its relations can be used for ontology representation. Several languages were developed for the promotion of knowledge representation and sharing as well as for data integration such as RDF [RDF], DAML+OIL [DAML] or OWL [OWL]. Currently, the mostly used standard ontology language is OWL (Ontology Web Language). OWL is recognized by the Semantic Web community [W3CSe] as a best suitable for ontology representation. It is written in XML format that enables a common, well-defined and easily processable syntax but does not involve semantic into the date description by itself [Bray00].

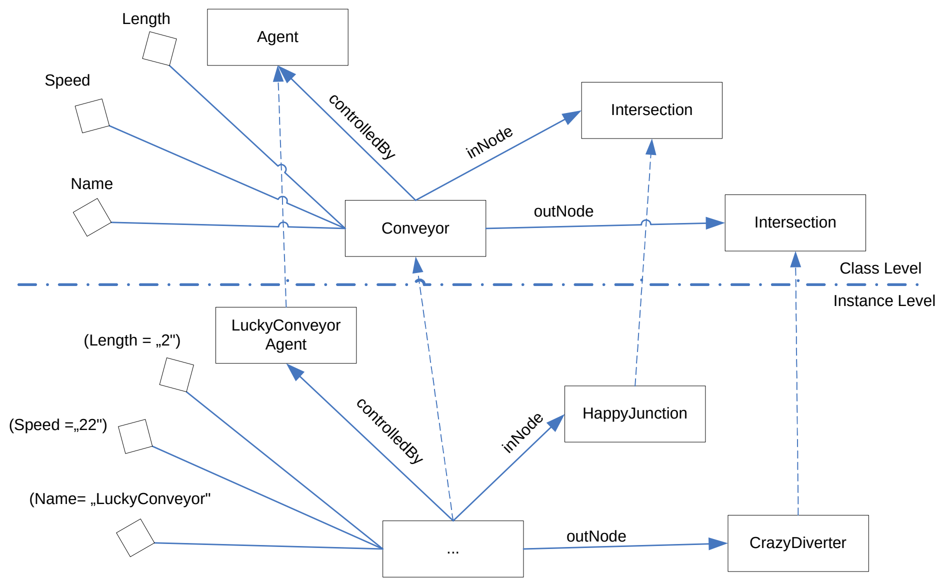


Figure 2.3: Conveyor relations and properties

For this purpose the Resource Description Framework (RDF) and Resource Description Framework Schema (RDFS) languages are used to describe interrelationships among resources in terms of named properties and values. OWL also supports the construction of distributed ontologies and offers the ability of accessing to a particular part of the ontology in order to update and revise it without interrupting the integrity of the top-level system ontology. OWL provides three increasing levels of expressivity in OWL Lite, OWL DL, and OWL Full respectively. This allows users to define their own needs for expressivity and chose a language version that supports their needs best. Due to its tight connectivity to RDF and since our approach is much more concentrated on the usage of rules for reasoning and less on Description Logic (DL), which can be used to determinate the semantics of OWL, as well as the decidability and complexity of basic inference problems, we will use the OWL Full for ontology representation in our approach.

Various ontologies have been developed to capture particular fields in the manufacturing domain:

- the OZONE ontology [Smi97] is devoted to constructing scheduling system,
- the Enterprise Ontology aims to define the overall activities of an organization [Usc98],
- the TOVE Ontology focuses on the enterprise modeling [Fox98],

- the “Machine Shop Information Model” is intended for representing and exchanging machine shop data, initially between manufacturing execution, scheduling, and simulation systems [McL05],
- Process Specification Language (PSL) covers generic process representation common to manufacturing applications [Grü05].

On the other hand, ontologies like MASON [Lem06] and the ADACOR [Lei04] could be classified as general-purpose manufacturing ontologies. An interesting standardization initiative has been started by the OOONEIDA consortium establishing the framework for both the hardware and the software interoperability at all enterprise levels. The product data, which encapsulates intellectual property along with appropriate semantic information, is collected from the manufacturer and integrators in order to set a searchable repository and ease the work of related intelligent repository agents [Vya05]. The complementary work has been reported by Lopez and Lastra, which merged separate ontologies for mechatronic devices reference models (covering both the hardware and the software features) and the IEC 61499 reference model respectively into the ontology for the Automation Objects reference model [Lop06].

Nevertheless, ontologies have been rarely used in combination with software agents and in most of the existing multi-agent systems the agents are not aware of ontologies at all since the information processing and reasoning are hard coded in the agents’ behaviors. Although important standardization work has been done by introducing the message transport service for sending FIPA-ACL Messages [Fou03] by defining message types, performatives protocols, etc., the agents are not able to semantically interpret the domain-specific content of the exchanged messages as well as the knowledge held by the other agents [Qiu05].

2.2.6 Knowledge Representation

In order to work effectively and efficiently an agent has to maintain an internal “world model” of the environment in which it is embedded and to keep this model sufficiently consistent with the real world [Woo86]. It means that the agent’s knowledge has to be accurately represented in the agent’s world model. In this content the knowledge representation is used to define the formalization of captured knowledge using a machine-readable form. There are three kinds of knowledge representation: tacit, implicit and explicit. Tacit knowledge is more personal, context-specific, and difficult to represent and explain since it contains skills, ideas, experiences, etc. Implicit knowledge is mostly embedded in the agent’s control and sensory processing algorithms. Knowledge is explicit when it is separated from the algorithm that uses the knowledge [Dav94]. The well known usage of implicit knowledge is in Brook’s subsumption architecture for the control of mobile robots [Bro86]. Presented reactive agents do not store any information about the world in their memory and decide what to do based on the current sensor values. Brook’s reactive agents quickly respond to changes, since there is no time consuming deliberation process and far less amount of information is considered. Nevertheless, the low flexibility of such agents in a dynamic environment, when they have to deal with incomplete knowledge, as well as their fragile problem solving abilities are mostly the result of the explicit knowledge deficiency. Moreover, in a dynamic, physical environment with real sensors, implicit representation can be computationally expensive and/or impractical since complete environment can not be observed at any time [Wass99]. On the other hand, although considered as complex and inefficient, the explicit knowledge is flexible and general, since it can be easily coded into information. The application of agents, which use the explicit knowledge for world model representation, can reduce the system complexity and enable system scalability especially through the interrelation of the agents’ world models.

Semantic systems offer a convenient way for the representation of distributed and interlinked explicit as well as tacit knowledge. The usage of ontologies for knowledge representation, sharing and high-level reasoning could be seen as a major step ahead in the area of agent-based control solutions [Obi02]. The exploitation of semantics and ontologies in the area of agent-based industrial systems has become the hot topic in the last few years due to the success and good promotion of the semantic web, which is the World Wide Web extension where the information is given well-defined meaning, to enable better communicate between computers and people [Ber01]. The ontologies are considered here as an essential technology for semantic web development guaranteeing data and information interoperability in such extensive heterogeneous and content rich environment. Related to its application in multi-agent systems, an ontology developed to describe the NovaFlex shop floor assembly domain and used as knowledge source for the multi-agent system is reported in [Bar05]. Two basic categories of concepts are proposed: modules and skills. The interesting application of an ontology developed for agent-based reconfiguration purposes is reported by Al-Safi and Vyatkin [AlS07]. The basic ontology concepts used here, are material resource and operation. We presented the application of an ontology in combination with multi-agent technologies in the transportation domain by simulating the assembly of products [Mer08]. The reported ontology introduces agent classes in addition to resources, activity and operation concepts. An ontology-based interoperability framework for the management of a distributed industrial manufacturing environment is proposed within the frame of the running PABADIS' PROMISE project [Kal07]. The proposed ontology aims to formalize conceptual information about resources, products and processes.

Summarizing the implemented test cases the significant improvements of the system due to the introduction of agent technology could be notified. Nevertheless, the deficit of real industrial involvements as well as applications with real-life scenarios is more than evident. The presented lacks of multi-agent approaches and missing standardization have been seen as main reasons for the weak deployment grade in the industrial community. On the other hand, these shortcomings could be beaten on their own through a massive application of this technology. In order to make agent-based systems more reliable and more attractive we notified several major challenges in section 2.2.4. In this thesis, we aim to face these challenges by combining advantages of multi-agent and semantic web technologies and test these on “real life” scenarios. The importance of such a symbiosis is especially seen through the way of how the information could be treated and managed in the distributed environment and how the knowledge could be used to improve the functionality and effectiveness of a distributed system. Having a multi-agent architecture that requires agents able to plan their action as well as to react in real time, we are applying a hybrid agent architecture that implies the usage of ontologies for representating the agent’s explicit knowledge and world model as well as the usage of the implicit knowledge embedded in the low-level control for actions that require reactions in real time. Since our agent uses behaviors that respond only on the world model state described with facts that could not have uncertain nature, the term knowledge used further within this thesis will be related only to explicit knowledge.

2.3 Planning and Scheduling

The production planning and scheduling issues are of essential importance for the manufacturing domain today, especially due to the dynamic and competitive nature of the nowadays global market that needs enterprises to be adaptive, flexible, robust and collaborative. In order to achieve this, the introduction of completely new approaches for problem solutions as well as more effective and efficient decision-making techniques is required. In this subchapter we will give an overview of the related work and suggest some directions for improvements in these fields.

2.3.1 Planning

Process planning (PP) plays a very important role in the product life cycle by linking the product design with the manufacturing phase. Process planning resolves between what and how will be produced. As a main goal of the process planning could be identified the full automation of the planning phase, so that the plan generation related to decisions, which tasks and in which order are going to be done as well as the corresponding operation and tool selection, occur without any external human intervention. The process planning phase has to consider the product requirements (price, quantity, geometry, tolerance, material, etc.) as well as production constraints (machine capacity, tool characteristics, etc.). The usage of computer technologies in process planning – Computer-Aided Process Planning (CAPP) – has made a significant step forward in the direction of full integration and automation of the design, planning and manufacturing phases in the computer-integrated manufacturing (CIM) environment. CAPP is commonly classified in two categories: the variant approach and the generative approach. The variant approach spreads all existing plans considering specific characteristics into related categories. In the case of a new product or quantity request, the most suitable PP will be selected and appropriately modified. It is a relatively simple and fast technique but principally done manually and dependending on the knowledge and experience of the workers. The generative approach applies knowledge-based systems and is usually combined with artificial intelligence (AI) techniques to generate the optimal process plan according to the part's features and manufacturing requirements. This planning process does not necessarily depend upon any other existing plan and is usual created automatically. The following steps in generative CAPP could be specified: product decomposition, part/feature recognition, interpretation of part design data, operation specification, machine selection, tool selection, determination of processing parameters fixture, setup identification and operation sequencing. Due to its nature to generate the "optimal" plans considering current system conditions the generative process planning system meets most of the needs of large companies which are dealing with a number of different products, each in small production sizes. However, in usage are mostly CAPP systems that apply either variant systems or semi-generative systems, since a truly generative process planning system that meets industrial needs and provides a reliable generic structure, knowledge representation model and reasoning mechanism is difficult to develop [Zha07]. Various CAPP approaches such as the object-oriented, genetic algorithm (GA) GA-based, Petri net-based, fuzzy logic, neural-network-based, feature-driven or knowledge-based approach have been reported in the literature and applied for solution of different problems in the process planning domain [She06]. The main shortcomings of those AI approaches are: knowledge acquisition difficulties, lengthy development time, slow and expensive in execution speed, and no existence of a general-purpose intelligence [Ber99]. Moreover, most CAPP systems applied today have a centralized and hierarchical architecture as well as off-line data processing that build all together a very inflexible structure and causes that plan regeneration takes a significant amount of time [Zha07]. Due to the lack of intelligent capabilities, such system have difficulties to automatically adapt plans according to the availability of resources, or share knowledge among the various planning related functional modules [Utp99]. Additionally, a low frequency of planning runs and difficulties in coping with new organizational forms of manufacturing such as product oriented or customer driven production, require new skills and new approaches capable to handle the shortcomings mentioned above [Aze00].

2.3.2 Application of Agents in Process Planning

As a possible way to overcome the shortcomings of the decentralized architecture that spreads the planning process between several entities/agents, each capable to create, control, and observe the execution of its own plans, is suggested. The agents cooperate and coordinate their actions in order to effectively accomplish their plans as well as to reach the commons system goals. Such organization brings time improvements, since the complex problems are partitioned between the entities by giving each entity a part of the problem to solve instead of dispatching the whole problem to the central unit, what could cause difficulties especially when the data is voluminous and changes frequently. The distributed agent-based approach allows proactive data processing at the place of its origin and data exchanges are only those necessary for effective system functioning [Hod05]. Moreover, large and complex problem structures become more simple and the possible failures easier to track [Sei03]. We present some of the successfully implemented agent-based process-planning systems.

An agent-based cooperative process planning system (CoCAPP) that integrates CAPP with CAD and CAM and demonstrates five major requirements – autonomy, flexibility, interoperability, modularity and scalability – is presented by Zhao et al [Zha00]. Six planning agents (P-agents) related to specific tasks in the planning process (such as: feature recognition, operation selection, machine selection, tool selection, etc.) are implemented. These agents are monitored by the B-agent that supplies the global state information of the problem and monitors the operational dependencies among the individual P-agents. The D-agent transmits the product design data from the CAD system. Another example of the applicability of multi-agent concepts for planning manufacturing processes is demonstrated at the Skoda Auto Engine Plant for planning their mass-production of car engines [Pec07]. The solution is based on the ExPlanTech multi-agent architecture [Pec02], which consists of a planning agent focused on the product configuration and creation of production plans for individual orders; managing agents in charge of detailed resource allocation and scheduling as well as conflict resolving and plan reconfiguration; and resource agents that are either the representation of factory hardware and software systems or that simulate a specific machine, workshop, or department. An approach to intelligent process automation, where a higher-level agent-based automation layer operates as a distributed planning and plan execution system that creates and runs reconfiguration sequences, is reported by Seilonen et al. [Sei03]. The agents form a hierarchy based on authority relations. Any of the agents that are representing physical or functional sub-processes can start the planning process. During the distributed and cooperative planning process each agent creates locally its own part of the overall plan and adapts it to the other agents' plans as well. The National Institute of Standards has developed an agent-based platform that supports the integration of predictive models, process planning, and shop floor machining activities [Fen04]. The agent platform includes a design agent, a group of process planning agents, the capability repository agent, and the manufacturing control agent. The agents have access to a knowledge base, a manufacturing resource database, a numerical control programming system, a mathematical equation solving system, and a computer-aided design system. The presented multi-agent system has demonstrated an approach for system interoperability and the optimization of process performances. The conducted and extensive literature reviews related to agent-based collaborative process planning were done by Zhang et al. [Zha07].

2.3.3 Planning and Scheduling in the Assembly Domain

Constraints set during the product design phase influence the definition of the process planning phase as well as the entire system design in a great manner. On the one hand, in order to automate the process planning generation, the product model representation has to be made in a way that enables understanding the designer’s intention, offer information about specific features (connections, definitions, constraints, etc.) that could be used for the selection of appropriate equipment as well as tool set-up definitions. At the same time, knowledge about the capabilities of the equipment could facilitate the product design and ensure its manufacturability. The ability to present the product model in a same way as production process and production equipment can support easier mapping between these three key manufacturing elements and enable easier optimization of both planning and execution process.

Assembly is much more than a process where two or more parts are connected, since the whole process is accompanied with preceding as well as following actions (supply, transportation, inspection, handling, delivery, etc.). All these actions are linked and there is a high amount of information that has to be exchanged between the actors in order to have them accomplished. “Assembly model” or models must be capable of capturing a diverse set of information needed to describe the entities and activities associated with assemblies and assembling so that designers of products, assembly systems, logistic systems, supplier relations, field support, and finally disassembly and recycling, can have access to the information they need [Whi96]. However, the lack of ways to standardize and describe assembly domain knowledge is an obstacle to achieve an easy flow of information. This is the reason why we select the assembly domain and its automation as test case for our knowledge-based multi-agent concept.

Vos elaborated major issues related to the relevance of assembly and assembly automation to the industry. He also notifies the importance of methods that support the configuration of assembly systems according to a product range and the production parameters [Vos01]. We modified his graph (Figure 2.4) that shows the link between product design, assembly processes and assembly equipment by considering also the influence of a customer order on the assembly process and recently more and more on the product design (customized order) .

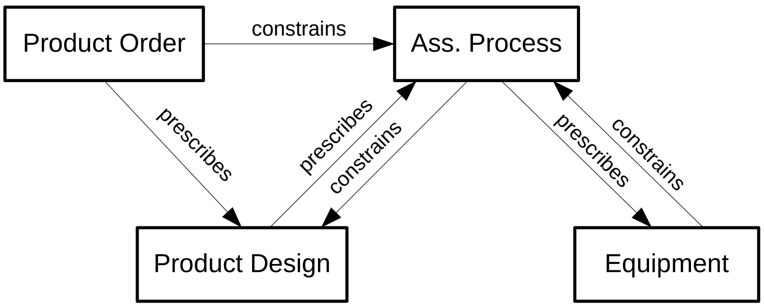


Figure 2.4: Link between product design, assembly processes and assembly equipment [Vos01]

The product order, labeled with the product type (though this is related to the product design), deadline and quantity, sets the key borders to the production planning process impacting directly the resource exploitation. This is the main reason why the product order parameters should be "understandable" and presented in the whole production chain, from the order over production until the final delivery.

On the other hand, also the large amount of relevant data emerging from the product development process (material, shape, structure, etc.) has to be coordinated and integrated in the manufacturing process and through the whole enterprise. This especially, while in order to reach a sustained product development, which is seen as essential step to achieve competitive advantage nowadays, manufacturing systems are required to manage the product throughout its entire lifetime ranging from design, manufacture, operation and destruction by establishing a collaboration between partners from a wide spectrum of domains, resulting in various product data types and formats, as well as different software tools [Mos05]. However, traditionally the product model is geometric based and provides incomplete product definitions because besides the assembly geometry, the understanding of its physical effects as well as the design intentions (e.g. joint type) is required. The meaningful representation of product data is necessary to enable semantic interoperability across different application domains [Pat05]. Ontology technology sets clear relations among assembly components and forms features by systemizing assembly knowledge in product, feature, manufacturing, and spatial relationship classes and linking data in a way that enables automatic reasoning as well as their wide integration. Kim et al. presented the ontology-based assembly model that serves as a formal, explicit specification of assembly design making assembly knowledge both machine-interpretable and shareable at the same time [Kim06]. They classified assembly knowledge into a hierarchy of assembly/joining concepts, defining concepts embedded in assembly design using specific terms and creating a standard vocabulary to describe assemblies (e.g. Assembly, Part, Sub-assembly, Assembly Feature, Form Feature, Joint,

Joint Feature, Mating Feature, etc.). The ability of the presented ontology to capture both assembly and joining intents was demonstrated with a realistic mechanical assembly. On the other side, in order to link products, assembly processes and assembly equipment, Lohse et al. proposed an assembly process ontology [Loh05], which could be seen as an extension of the PSL ontology presented before. In contrast to Kim, whose ontology concept covers product design and development side, this ontology provides definitions of concepts which are relevant for the specification of an assembly process including a detailed product and process model. The structure of an assembly process is separated into three levels: tasks that define the sequence in which the components are assembled to form the final product: operations that define the steps required to put the components together; and actions that define the individual motions and other hardware and control related activities. The product is defined within this concept as a hierarchy of assemblies, components and parts linked together by liaisons whose types specify the choice of appropriate assembly operations. Both ontologies, although built with other intentions, have some mutual classes (e.g. Assembly, Part, Joint/Liaison, etc.) that could be used as merging points for the integration of these two concepts covering a broad range of product related applications. There are also some other approaches related to assembly modeling and process planning proposed in the literature [Zha03].

"*The use of ontologies and explicit semantics enable performing logical reasoning to infer sufficient knowledge on the classification of processes that machines offer, and on how to execute and compose those processes*" [Las06], what makes them proper for the fusion with agent technology and to be used in the planning process. In order to reach the goal of modern assembly planning systems to create activity sequences that are not only feasible but also optimized according to one or more parameters, such as makespan, machine or tool utilization, the agents that are supervising a particular resource or process planning system can use the accurate information stored in the ontology to reason about available resources and utilize appropriate optimization heuristics [Las06]. To our knowledge there is very few research in the area of agile agent-based assembly systems. A framework for an agile assembly system has been proposed in [Hol95]. The ongoing work on knowledge-based automatic reconfiguration system for robotized work-cells, where the problem of reconfiguration is seen as a (re)planning problem and knowledge representation is built around the concept of ontologies, was presented by Malec et al. [Mal07]. The development of an ontology-based reconfiguration agent that uses ontological knowledge of the manufacturing environment for the purpose of reconfiguration without human intervention is reported by Al-Safi and Vyatkin. The created ontological knowledge model of the manufacturing environment is based on the MASON ontology and used by the configuration agent to infer facts about the environment [AlS07].

The application of agent technology does not bring any advantages if the used agents are not intelligent. Considering ontologies as an intelligent way to manage knowledge, the integration of both technologies brings advantages such as extensibility and communication, enabling agents to agree on the meaning of common concepts they use with any other agent in an open environment [Gon06]. The application of both technologies on the concept that covers the entire life cycle of a product is required.

2.3.4 Production Scheduling

The scheduling of production resources is one of the key features in the current competitive and dynamic manufacturing environment. The scheduling has to be flexible and able to cope with conflicts derived from the resources shared among the production orders. The scheduling determines the most suitable time slot to produce something. The scheduling strategy that can support a fast reaction to market changes and can cope with a turbulent environment is considered as a one of the key issues in such systems. The task of scheduling is the allocation of jobs and activities to available resources over time considering relevant constraints and requirements [Raj06]. Its main objectives are the minimization of the production time of jobs, production costs, increased resource utilization, etc. Substantial research efforts have been devoted to developing mechanisms capable to dynamically allocate the resources required to support the production activities needed for fulfilling the order [Won06, Pin02]. However, most of the developed scheduling systems are based on centralized structures, which make manufacturing systems scheduling even more complicated. Due to the hierarchical and centralized structure, their modification is a very expensive and time consuming process. This is especially obvious in the cases when certain resources become unavailable or additional resources are introduced to the system, since their rigid structure hinders flexible redesign of the system. In most real-world environments, scheduling is an ongoing reactive process where the presence of real-time information continually forces reconsideration and revision of pre-established schedules [Oue07]. The centralized structures, based mostly on big and complex databases modeled at one (central) location, often suffer from data consistency problems and are not always capable to carry out this dynamic rescheduling. This especially, in the cases when the system has to manage data and knowledge which is spread between different locations, since the different coordination and communication mechanisms are required in order to keep the system synchronized. Highly dynamic and complex systems, such as manufacturing systems are hard to manage with such structures. In the past several different analytical and heuristic methods (including GAs, fuzzy logics, neural networks, Tabu search, etc.) were applied in order to cope with scheduling problems and their optimization. However, being essentially centralized and based on simplified theoretical models, such methods have shown difficulties when applied to real-world situations [She06].

The application of decentralized control architectures is considered as a promising approach to overcome the issues mentioned above. Multi-agent systems, which are based on this approach, are handling the complex problems by breaking them down into a number of simpler problems, which are distributed between agents that then join their efforts in order to produce a solution for the global problem. Agents cooperate and communicate together in order to achieve aims, which are beyond the individual capabilities and knowledge possessed by each agent. Nevertheless, being able to utilize parallel computation and to apply different methods for solving their simple local problems, the application of agent technology can significantly improve the efficiency and performance of the entire system. The agent concept is characterized by the application of a bidding mechanism that in contrary to time-based mechanisms introduces the economic-based evaluation or combination of these two principles [Ush03]. The Contract Net Protocol (CNP) is the mostly used bidding mechanism proposed by Smith [Smi80]. The agent that needs some service broadcasts its tender to other agents and waits for their bids. After a particular time, all received bids are evaluated and the best provider awarded. In order to enable negotiation among the agents to form different types of coalitions, an extension of the CNP based on an market-oriented approach is proposed [Wel95]. The CNP is combined with dispatching rules to solve job-shop dynamic scheduling problems [Yin07]. A further advantage of the agent based scheduling is its capability to handle dynamic system changes (e.g. machine breakdowns, rush orders, etc) [Boc04]. Being applied in a distributed environment, agents need only to “freeze” the part of their schedule and recalculate tasks that are related to the broken resource bringing the system due to their separated actions into a new evolved state.

Two main multi-agent architectures for dynamic scheduling are reported in the literature: autonomous architectures and mediator architectures [Oue07]. The autonomous architectures are usually based on negotiation between heterogeneous units. Parunak et al. presented the AARIA agent architecture, where completely autonomous agents supervise physical components (machines, tools, transportation resources, etc.) [Par01]. Saad et al. adopted a multi-agent cooperative problem solving paradigm and presented heterarchical scheduling approach using a bidding mechanism based on the CNP to generate the production plan and schedule [Saa97]. Various autonomous architectures for dynamic scheduling in flexible manufacturing systems were also reported in [Kro99, Gol98, Oue98]. The main advantages of the autonomous architectures are reduced complexity, integrity, cost efficiency, high flexibility, and a high robustness against disturbances. The lack of predictability and global perspective are major drawbacks of this architecture. This kind of architecture is also characterized with an increased amount of communication, since the agents have to exchange more messages in order to have an accurate environment representation. Shen and Norrie proposed the mediator architecture consisting of various physical agents, which use the services of a mediator agent to coordinate their activities [She98]. There are also few other approaches described in [Bon97, Sun01, Won06]. The advantage of scheduling through mediation is that the mediator possesses sufficient knowledge about its coordinated agents and can coordinate their activities ensuring therefore global consistency. The main drawback of this architecture is its “centralized” structure with mediator agent as its bottleneck, whose failure can cause a failure of the whole system.

Extensive surveys of dynamic scheduling in the manufacturing environment considering also agent-based systems were done by Babiceanu and Chen [Bab06] as well as by Ouelhadj and Petrovic [Oue07].

2.3.5 Integration of Process Planning and Scheduling

Process planning and scheduling are highly related, because when the planning ends the scheduling phase starts. However, being mostly done offline the process plan generation often does not consider the current status of the shop floor. There are a number of factors that could initiate uncertainties in the production system such as the existence of complex and reentrant products, unreliable machines and stochastic yields, where small disturbances lead to a disruption of the manufacturing process, often causing order disruptions and consequently changes in the plans and delivery problems [Aze00]. The process plan restrictions and shop floor constraints have to be considered in the scheduling phase that could become a very complicated and time consuming process, if applied in a dynamic environment. In order to make more realistic and applicable plans, the integration of the planning and scheduling phase is necessary. Nevertheless, the traditional approaches execute these processes separately, mostly ignoring the condition of resources on the shop floor (e.g. machine workloads, etc). That leads to the under- or over-utilization of certain resources or even that some of the process plans perhaps cannot be executed requiring alterations or replanning [Kum06]. A lot of work has been done in the past to optimize and integrate process planning and scheduling in the area of manufacturing and the resulting approaches can be classified into the following categories: centralized optimization algorithms, close loop optimization, distributed process-planning approaches and agent-based approaches [She06]. In the agent-based approach, all related actions are done by agents that are capable to communicate, negotiate and accomplish specific tasks. An integration system that consists of a manager agent, a process sequence agent, a machine grouping agent, a scheduling agent, an optimization agent, a number of product agents and various resource agents is presented by Lim and Zhang [Lim04]. Agents are classified into two categories: execution agents and information agents: and each agent has two types of objectives: global objectives (e.g. to integrate dynamic process planning and scheduling, to dynamically optimize the utilization of manufacturing resources, etc.) and local objectives (e.g. bid and win the jobs announced by the machine grouping agent…). Wong et al. presented an approach where the selection of the schedule and allocation of manufacturing resources is achieved through negotiation among the part and machine agents by using hybrid CNP (negotiation on a fictitious cost with the adoption of a currency function) [Won06]. Two MAS architecture, a simple one consisting of part and machine agents and a hybrid-based architecture that involves in addition a supervisor agent, were tested demonstrating the ability of the hybrid approach to provide solutions with a better global performance. Tehrani et al. presented a multi-agent architecture of an integrated and dynamic system for process planning and scheduling for multi jobs. The alternative manufacturing processes are presented by the process plan networks and their heuristic search algorithms are combined with the negotiation protocols, in order to generate suitable process plans and schedules in the dynamic manufacturing environment [Teh08]. A comprehensive state-of-the-art review in the area of manufacturing process planning and scheduling integration was done by Shen et al. [She06].

However, one of the main shortcomings of the architectures mentioned above is the lack of interoperability, since the applied methodologies separate planning activities (e.g. process planning) from executing activities (e.g. production control and scheduling), creating a gap between the involved systems. The problem in current distributed systems is that they are still tightly coupled from the point of view of automated gathering and integration of data, information and knowledge, being programmed with the focus on performing particular tasks rather than on interoperability and openness [Obit08]. Shen et al. defined the integration of process planning, manufacturing scheduling, and control as one challenging research topic where much more attention has to be set on the complexity analysis and formal modeling of such integration [She06]. The assimilation of different knowledge sources is considered as an important problem that has to be solved being marked as not easy task due to different representations, foundations, and levels of abstraction of various knowledge sources [Bos99]. Being mostly applied in heterogeneous environment, agent has to understand its as well as the knowledge of related agents in order to reason about it, prior to making decisions. Moreover, considering that future distributed manufacturing systems will need to handle a great diversity of autonomous agents and mechatronic devices interacting intensively, there is as strong need that all components understand the exchanged information and know how to communicate [Chr07]. According to Finin et al. for software agents to interact and interoperate effectively three fundamental and distinct components: (i) a common language; (ii) a common understanding of the exchanged knowledge; and (iii) the ability to exchange whatever is included in the previous two; are required [Fin97]. The usage of machine-interpretable semantics (ontologies) to describe the components of manufacturing systems enables other intelligent components (agents) to perform reasoning and infer sufficient knowledge to interact as well as to overcome current interoperability barriers [Las06].

This is the reason why we used the semantic technology in this thesis to solve the interoperability problem in the multi-agent domain, solving directly and indirectly the interoperability problem in all the domains (including control, planning and scheduling) which are managed by agents.

3. The KASA Environment

"Beauty is in the eye of the beholder, and information is in the head of the receiver."
Dretske

3.1 Introduction

An agent is an intelligent entity placed in particular environment in order to supervise or execute specific actions. It is able to perceive the environment through sensors and act on it with effectors. Based on its responsibilities and observations, an agent has to constantly make decisions that again could influence the environment as well as its state. An agent has to be aware of the results of its actions in order to be able to perform subsequent activities or repeat the whole process over again (Figure 3.1). Nevertheless, while its actions could have beneficial as well as negative effects to the agent's existence, the fundamental question here is: how should such an agent decide what to do? As possible answer, here should be considered facts and states required to start particular actions. The real and accurate information about the environment could be considered as key precondition, needed by the agent in order to make proper actions efficiently and effectively. Consequently, for an agent it is of crucial importance to "understand" received information from its surroundings and related to its meaning select correlated behavior that will influence environment over effectors.

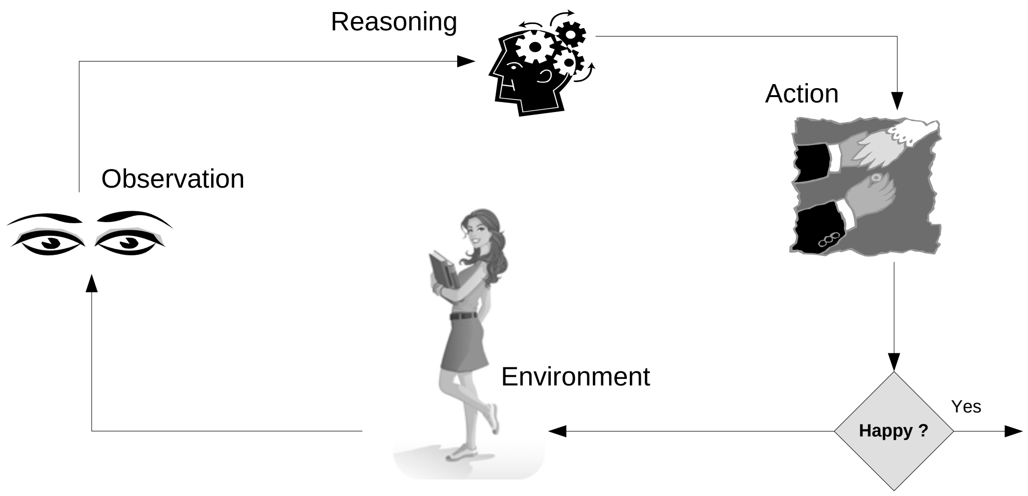


Figure 3.1: Correlation of the agent and its environment

This indicates the importance for an agent to mirror and transform its environment into an understandable mental representation. This representation is called world model and can be defined as the agent's internal representation of the external world or its domain of application. The representation is, within this context, defined as relationship between two domains where the first is meant to "stand for" or take place of the second, being more concrete, immediate or accessible in some way than the second [Bra 04]. Such a world model represents objects, activities, and states embodying the agent’s knowledge about its surroundings. As said in the previous chapter, we will use an ontology to formalize the agent’s knowledge. Consistent semantics assure the accuracy of each agent’s world model with respect to the real world. Providing an ontology of its surrounding the agent is equipped with an up-to-date representation of its environment built from sensor data and communication with other agents. From the viewpoint of inter-agent interactions, the explicitly defined and commonly accepted ontology is an indispensable tool for ensuring interoperability between agents in the sense of providing the agents with common understanding of the knowledge exchanged during inter-agent communication. An ontology can also capture actions and events in a uniform and processable way so that they can be recorded in time and further analyzed. However, being used for diverse tasks agents have different structures of their world models. This depends mostly on their application domains but is also strongly influenced by the agent decomposition approach (physical or functional). Consequently, depending on their application domain, agents will have different capabilities and responsibilities. To define an agent, certain facts need to be specified about the agent: domain of application, tasks, environment, perception abilities (i.e. ways to update its world model: sensors, communication…), as well as agent behaviors in particular situations.

A system is defined as “*objects and events connected and controlled in time and space in order to obtain intended functions*” [Hol06]. Considering a multi-agent system as a set of related entities integrated in a complex society, where each entity has to follow particular norms and use specified mechanisms to interact with each other pursuing its own goals, there are particular system characteristics (e.g. stability, security …) that have to be respected in order to achieve a common system goal. Moreover, the agent technology embodies a large set of decisions and interaction capabilities, which are able to create a vide variety of system behaviors and when not designed carefully could lead to unintended behaviors or to the achievement of intended behaviors in an inefficient way [Bus04]. This all indicates that particular system states which are related to the global aims have to be integrated and considered when designing the agent’s world model. A manufacturing system, which is made of subsystems that could be also complex systems themselves, being mostly influenced by extremely turbulent conditions and characterized with a high number of system states, is a typical example of a complex environment. The introduced multi-agent framework, used to manage particular parts of this environment, should be able to handle the complex dynamics of the manufacturing environment allowing its members to coexist and perform activities that should bring this system into an optimal state related to its current conditions. Having a multi-agent architecture which is applied in the manufacturing environment to the assembly domain, we will analyze the manufacturing environment and address its relation to the assembly process in the next section. We will define agent types needed for its proper functioning, addressing also responsibilities and activities of each agent type. At the end of the chapter the developed agent architecture will be explained in detail. It is important to mention that the developed multi-agent architecture needs to satisfy requirements of the assembly domain, but should be applicable to most of the manufacturing environments.

3.2 A Manufacturing System

A manufacturing system is defined as *"a collection or arrangement of operations and processes used to make a desired product(s) or component(s)"* [Bla91]. It consists of interrelated elements (people, equipment, sub-systems, etc.) introduced to cooperatively achieve the overall objective defined as transformation of raw material into commercial products. Manufacturing systems could be categorized according to the type of production process to: *discrete manufacturing* that is concerned with the production of solid products and the *process manufacturing* that is related to the production of shapeless materials. Further, discrete manufacturing systems can be classified based on their production quantity to:

- *mass production*, which is used for producing or processing of extremely large volumes products without interruption and characterized by a production that runs permanently being executed by a series of machines that receive materials through a closed transfer system,
- *batch production* is meant for the production of medium size quantities of one type products or parts (shoes, books, inks, furniture, etc) having products produced in regular intervals, but with a production rate that is usually higher than the demand rate (in order to produce the next batch the equipment mostly must be stopped and re-configured causing inefficient loss of production time known as 'down time'),
- *job-shop* production that is characterized with small or very often size-one product volumes, being mostly adjusted to specific customer requirements and able to produce a wide range of products.

Within the production process one can also distinguish operations which are adding value to the product such as:

- *processing* where the properties of material are changed, and
- *assembly* where several product parts are combined into one.

There are also operations that add no value to the product but have to be performed in order to set preconditions for other operations (handling, fixing, etc.) or to test results of particular operations (inspection) [Bus04]. The processing and assembly operations are often interlaced during the production process and looped until the final product is being reached (Figure 3.2). Nevertheless, manufacturing systems are not only related to the production process but also include and manage subsystems that influence the production such as ordering, supply, shipment, etc (Figure 3.2). In addition, we noticed the importance of control, planning and scheduling subsystems in the previous chapter. Each of the subsystems mentioned above has to be supervised and its data, states and parameters considered, with much of this information needed not only in their original subsystem, but also in the other subsystems, such as e.g. quantity information is required through the whole production chain.

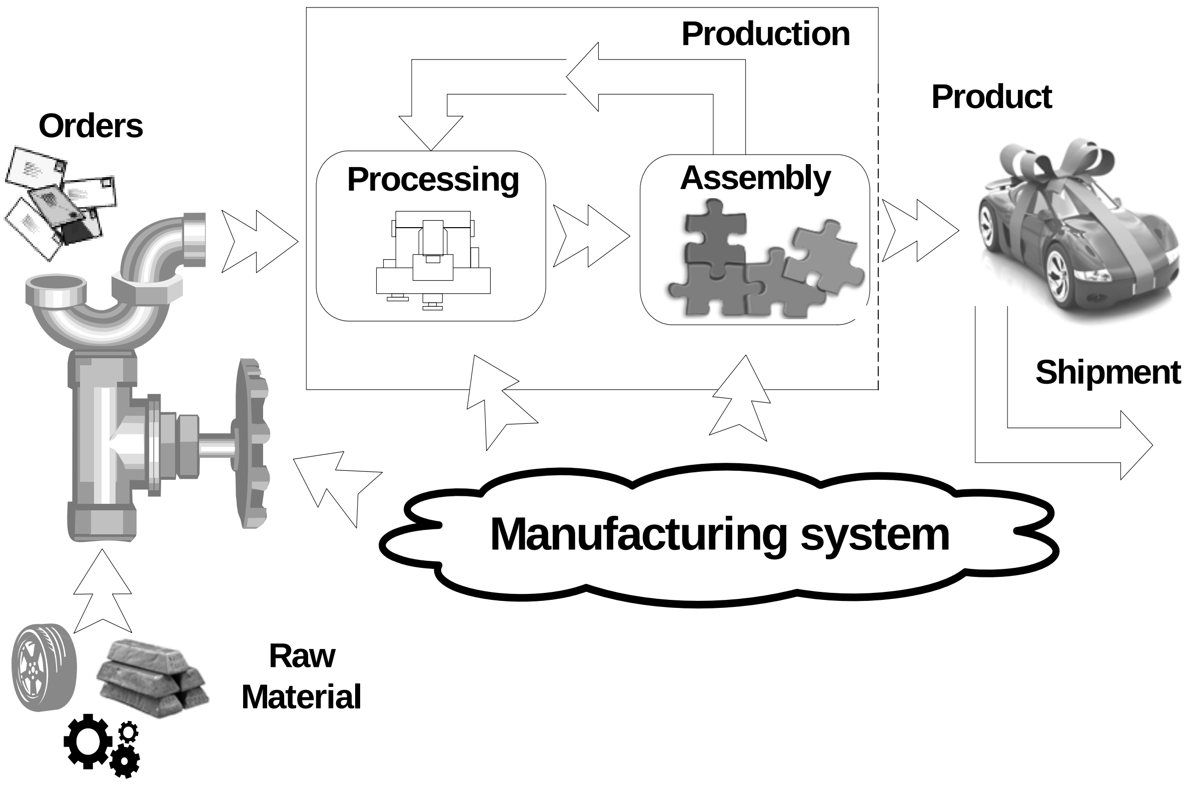


Figure 3.2: The manufacturing system

This becomes more complicated when the number of products, services, subsystems, and cooperation partners increase. Adding to this, requirements on manufacturing system to react quickly and competitive to customer request, to maximize resources utilization, to rapidly absorb system failures or to rash adapt their configuration on new products, etc. one can see how complex this system is and how a disturbance or an unexpected change in one part of it can affect any other part or even the whole system. A better coordination between subsystems would reduce the uncertainties and ensure making proper decisions. This points out the vital importance of an uninterrupted information flow between the manufacturing subsystems as well as their easy integration and their ability to “understand” this information, since this can improve the manufacturing system agility and its real time responsiveness. The KASA is designed to support the job-shop production covering initially the assembly processes but being able to support processing as well. The architecture is focused on a clear decentralization of the manufacturing system intending to reduce its complexity and increase agility.

3.3 Layered Manufacturing System Architecture

In order to reduce its complexity, the control of a manufacturing system can be spread in several “hierarchically” ordered layers. The layered system structure enables its functional decomposition into subsystems, which can then be easily further decomposed but also integrated and managed in bigger systems as well. Moreover, the entities within particular layer can work independently to a certain point and their failures do not necessarily have to affect the other neighboring entities or the whole system. During the layered structure specification, besides the fact that particular subsystems logically symbolize some layers (planning - planning control layer), we considered that particular decisions have to be made in a real time requiring observation of a limited environment in contrast to decisions that require a global view without a special time limitation. The introduction of layers limits their responsibility and planning perspective, improving system performances and simplifying the concept.

We specified four layers: management, planning, scheduling and executive layer (Figure 3.3). Their functions and responsibilities are listed as follows:

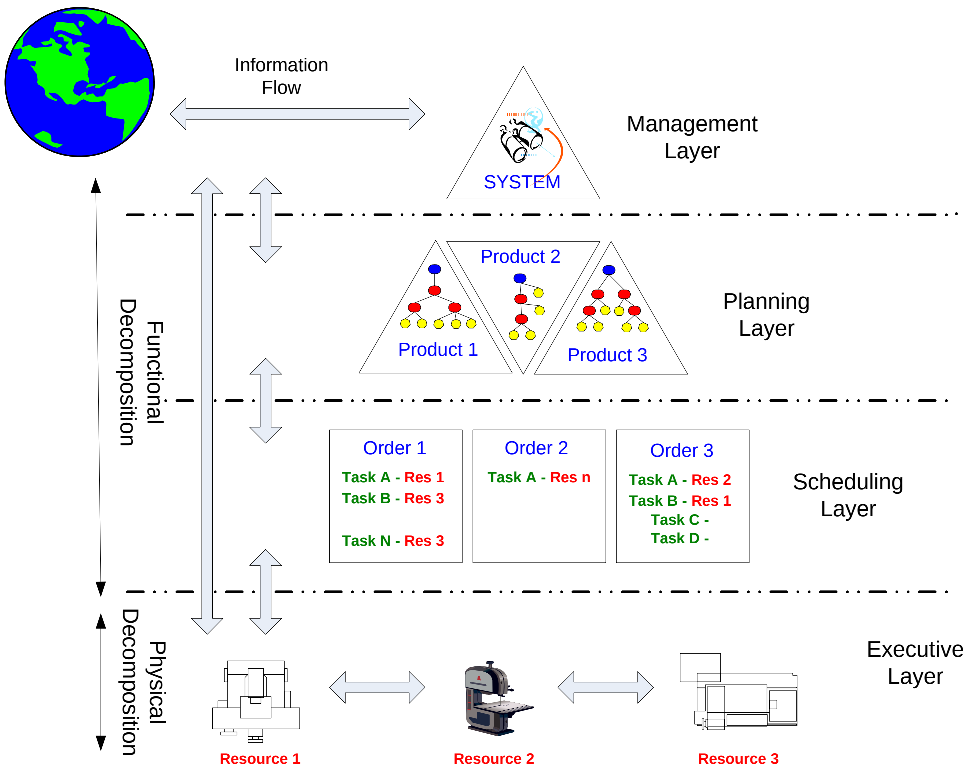


Figure 3.3: The layered system structure

- *The Management Layer* is responsible for entire system stability and functionality. It supports production and resource initialization as well as their determination. It is also concerned with the communication with the external environment and provides solutions for complex problems related to the global environment. It accepts orders on a routine basis.
- *The Planning Layer* links process planning with product design. It is basically concerned with the sequencing of process steps, identification of product types and quantities to be produced. It defines equipment and resources that could be used and ensures that the parts or components required for the production are available and the final product delivery dates not exceeded. The shop floor layout is also defined on this layer.

- *The Scheduling Layer* is concerned with the synchronization of production needs with available resource capacities. The goal is to reach the internal deadlines that are set on the planning level. This layer is responsible for negotiating with the resources, the tasks as well as parts, tools, and product allocation between resources.
- *The Execution Layer* is related to the physical job shop equipment. On this layer, the production tasks are executed considering the resources' constraints and abilities, their performances measured and if a failure or disruption is diagnosed, the scheduling as well as management layer is informed. Also specific activities related to the execution of particular actions (i.e. pallet routing, removing, fixing, etc.) are coordinated on this layer.

Such layer structures enable that related layers can communicate avoiding the unnecessary "stage by stage" procedures. The presented structure assures clear definitions of each layer role in the system as well as associated tasks that have to be done in order to achieve common goals. This further enables the smooth creation of related agent classes and mapping of ultimate system goals to these agents. Moreover, in order to enable easier decomposition and structuring of manufacturing goals and their linkage to related agents classes, we split one customer order in four layers. An *order* is composed of related *product orders*, which further consist of several *work orders* where each work order require execution of connected *tasks* for its accomplishment (Figure 3.4).

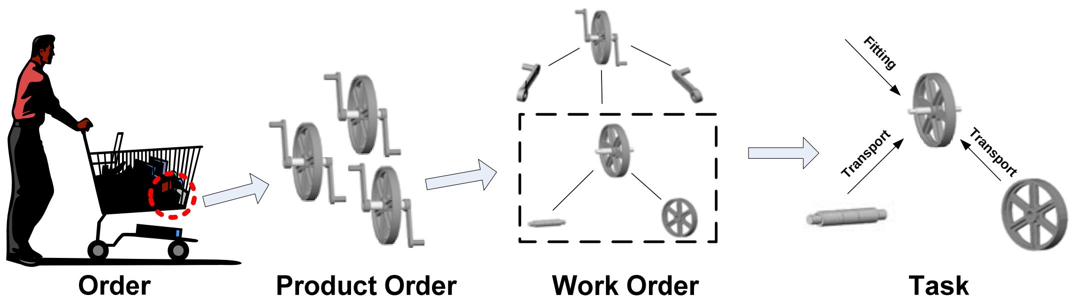


Figure 3.4: Order decomposition

In this context, *order* represents a set of products ordered by a customer, which specifies due date as well. *Product order is* defined by the type of the product, its quantity, design e.g. color, etc. *Work order* integrates all tasks that have to be done to make one subassembly for a particular product. Each operation done within one particular *product order* is called *task* (e.g. transport of a first part from storage, transport of a second part from storage, as well as the welding of these two parts are three tasks to be done to make one subassembly). The managing of each decomposed order layer is coordinated between particular agent classes. In the next section, we will present the resulting multi-agent system.

3.4 The Multi-agent System

We developed a framework for agent systems based on the hybrid structure, in which every agent can influence any other agent's behavior when needed and where each agent manages its own activities based on its local state or on the information (message) received from other agents. As a result the system behavior is not only reflected by the skills of one agent but also evolves from the collective behavior beyond individual agents. In the presented architecture, the agent is an autonomous semantic entity having specific tasks and knowledge about its domain of application, about strategies that can be used to achieve a specific goal, and about (other) relevant agents involved in the system. Considering the manufacturing domain as a complex system whose proper functions can be represented with a mixture of physical and non-physical components as well as relations between them, we combined both functional and physical decomposition approaches to create agents. The manufacturing components are “agentified” to implement a behavior that represents the manufacturing component objectives and functionalities, with each agent being responsible for carrying out different specific functionalities. We used the functional decomposition approach to create agents responsible for system support, process modeling and task scheduling (Figure 3.3), applying this approach to each of the first three layers and generating particular agent types for each layer.

The ***Contact Agent*** (CA) is related to the Management Layer and according to that it has responsibilities that encompass organizational and supervisor functions. The CA is created at the start-up of the system and it is always active. It is concerned with the system stability and in the case that one part of the system collapses, this agent considers its influence on the system performance and, if significant, undertakes particular steps in order to bring the system back into the optimal state. Its further responsibilities are to receive a customer order and create one Order and Supply agent for each related product order. This agent also creates an agent for each new resource introduced in the system. After the order was accomplished or particular resource removed from the system, the CA determinates the related agent. However, having only one instance of this agent for the whole system and considering it as a possible single point of failure, the replication technique [Mel05] and replication service provided with used agent platform [Bel07] can be applied to enhance agent’s failure tolerance level (more about the system failure tolerance in the Chapter 5).

The ***Order Agent*** (OA) captures the goals and tasks of the Planning Layer. The OA is responsible for accomplishment of one product order, respecting due dates and the like; and handling customer requests for modifying or cancelling their orders. The essential information for an order agent is: type of product, the production deadline, quantity, and the priority of the client. Having the knowledge about all products corresponding to a single order, this agent combines the ontology-based model for a particular product together with other information, sequences this into work orders and sends it to the supply agent. Based on this knowledge and contacting the storage, the OA checks if all parts and materials required for execution of a single order are available. During the production, this agent collects also information concerning the status of current product orders or the system’s performance. The OA is responsible for loading products into the system when a product order reached the system and for unloading products from the system when all of their processes are finished.

The ***Supply agent*** (SA) maps the functions of the scheduling layer and has such name since it “supplies” the job shop with tasks. The SA is in charge for coordinating the production execution in order to achieve the best possible production results, including on-time delivery, cost minimization, and so forth. It also manages the movement of related product order’s subassemblies and materials across the job shop. After the OA decomposes the product order into work orders, they are forwarded to the SA. Using the ontology and taxonomic relations specified in the product definition (Figure 3.16), the SA extracts tasks from work orders and schedules the ones that have to be completed at first. After that, the SA initially sends requests for bids to all machine agents that have the capability to process the first task. The interested machine agents respond with their bids. Each bid contains an estimated queuing time and finishing time for the requested operation. After collecting the bids, the SA evaluates the bids and selects the best one. When the related machine is identified, the agent negotiates with transport agents to route the task there. Whenever a current task is completed, this agent sends bid requests for the next operation. This bidding procedure continues until all the requested features of a job are finished. When the last task in the production process is finished, the agent sends the notification to the OA.

The physical decomposition was used to create the ***Machine Agents*** (MA), which are related to the Executive Layer. MAs represent manufacturing resources (typically a machine) providing particular processes und services. They play an important role in the system since they are the “hard workers”, whose effective functionality ensures normal system functionality. These agents have knowledge about their particular domain of application. The provided services by the resource are registered in Directory Facilitator so that the SA can easier find out suitable resources. The machine agent manages its local scheduling and negotiates with the SA about supply about free timeslots in which the requested operations can be performed. These agents collect the knowledge about all possible processes that can be provided, materials to be used, a list of geometrical features as well as the feature relationships, tools to be used to produce the feature and tolerance and surface quality requirements. A storage agent control reserves of products, parts and materials on the shelves. Machine agents update their knowledge permanently to ensure the existing record is kept up to date.

3.4.1 Testbed

As a testing platform we use the “Test-bed for Distributed Holonic Control” at the Institute for Automation and Control, Vienna University of Technology (Figure 3.5). The Testbed architecture consists of an automatic storage system with a handling unit for the extraction of the parts, a pallet transfer system with redundant paths, the industrial robot for machining and assembly tasks as well as a portal robot for the final assembly. Robots and the handling unit are considered as entities able to perform a certain operation.

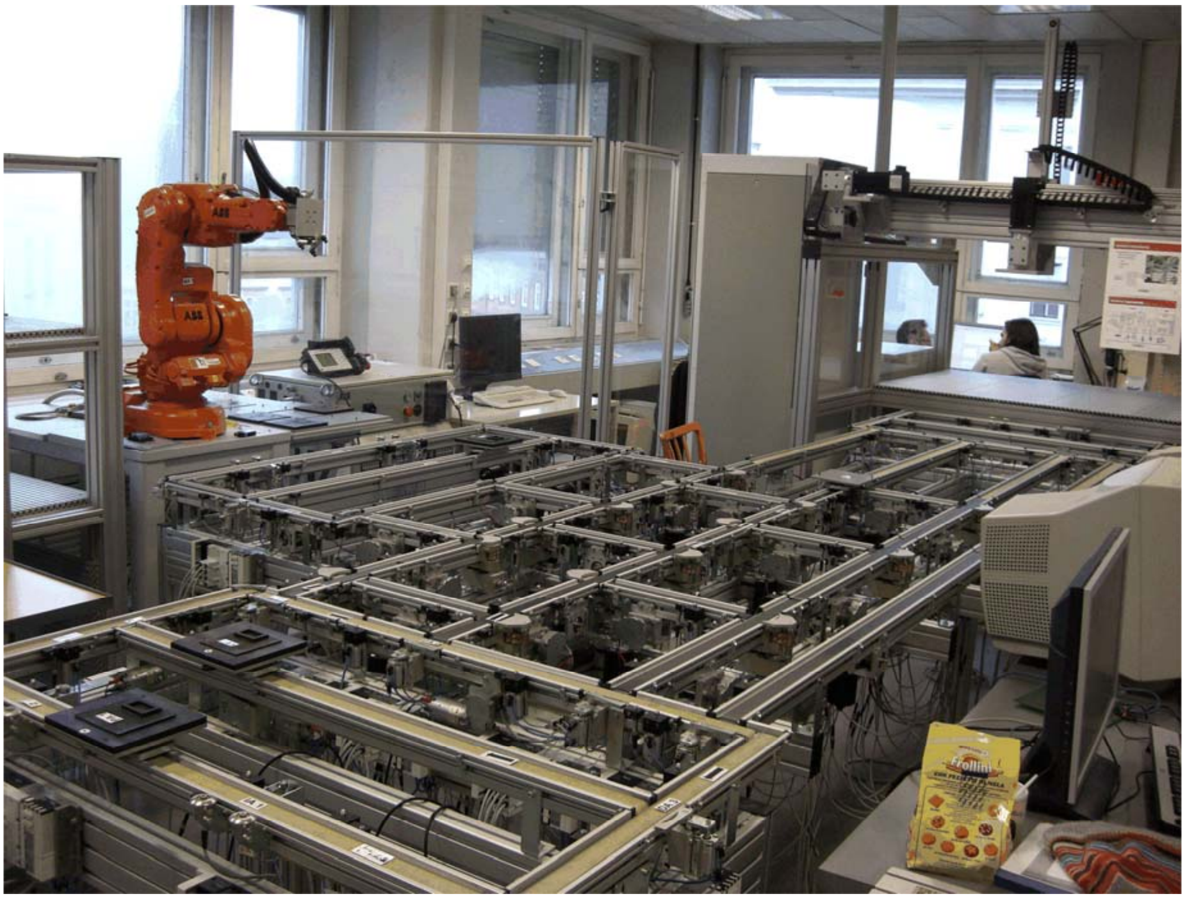


Figure 3.5: Test-bed for Distributed Holonic Control

Corresponding MAs register their services in the DF (will be explained later) during start-up. Each such production resource has an agenda, which is used to store the information about unfinished assigned tasks. Based on available knowledge the MA decides when and how a specific operation from the agenda will be performed. However, tools are also represented as a component required in order to perform particular operations and related to that the setup and the machining times as well as removing and handling times for each particular operation are covered in the ontology. The total processing time for one task is calculated as follows:

$$T_{task_{total}} = M_{setup_{time}} + T_{change_{time}} + (P_{machining_{time}} + P_{handling_{time}}) * Q \quad (3.1)$$

where $T_{task_{total}}$ is the tasks' total processing time;

$M_{setup_{time}}$ is the machine setup time for a particular operation;

$T_{change_{time}}$ is the tool change time;

$P_{machining_{time}}$ and $P_{handling_{time}}$ are part machining and handling times and Q is the batch size.

The total processing time for all assigned tasks $T_{agenda_{total}}$ is:

$$T_{agenda\ total} = \sum_{i=1}^{n} T_{task_{total}}(i) \qquad (3.2)$$

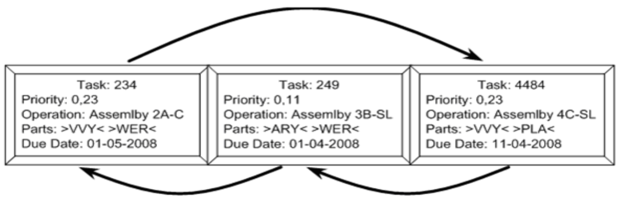


Figure 3.6: Priority based scheduling

where n is the number of tasks in the agenda. The MA controls the machine capacity and availability. The capacity limits the number of tasks that a specific resource can handle at a given time. Since each resource has a restricted capacity, the MA takes care that $T_{agenda_{total}}$ does not exceed the capacity. The MA has knowledge about each operation that could be performed at a machine and after it receives the Call for Proposal (CFP) message from the SA, the $T_{agenda_{total}}$, which also includes the time of the task from CFP, will be calculated and sent back as the bid. In the case that full capacity is reached, MA will not answer to the Call for Proposal (CFP) messages sent by the SA until its capacity become undermined or exhausted. However, when the resource starts with the execution of a particular task, its status will be set to busy and it will be not available for the next tasks until the previous one is completed. The MAs are using dispatching rules for sequencing the tasks allocated to their machines (the procedure will be explained in the next chapter). After the machine has finished the current task, the MA loads the next task that has the highest priority (Figure 3.6), if there is a confirmation from the Storage Agent (STA) that the parts required for that task are on the way. In this case the machine will be kept busy so that it is ready to process the highest priority job as soon as its parts arrive. Otherwise, the next task with lower priority but available parts will be loaded.

Components of the pallet transfer system and related agent classes

The main components of the pallet transfer system and related MA classes are:

- The ***conveyor belt*** which delivers items from one place to another. It is controlled by an agent that has to have knowledge about all conveyor characteristics (speed, length, direction, etc.). This agent also takes care that the number of pallets on the conveyor doesn't exceed the optimal number.
- The ***index station*** (Figure 3.7) that fix the pallet in a defined position for the handling units. Its agent is informed by the relating handling unit, which pallet to stop in a particular moment as well as when to release it.

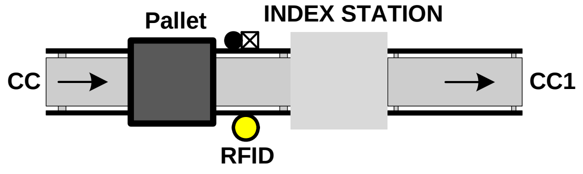


Figure 3.7: Index Station unit

- The ***identification unit*** (RFID) for the identification of passing pallet units. The agent sends this information to the related intersection unit.

- The ***intersection unit***, used as a common term for the place where three conveyors cross each other at the same level. Two different kinds of intersection could be deferred:
  - The ***Diverter*** (Figure 3.8) which receives items coming from the input conveyor and according to their destinations routes them to one of the two output conveyors. The related agent has to have an accurate system representation in order to be able to route the pallet correctly.

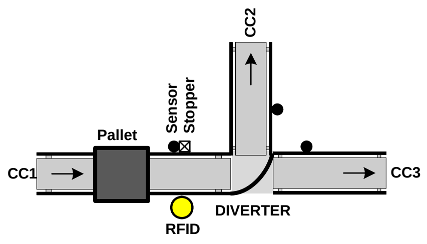


Figure 3.8: Diverter unit

  - The ***Junction*** (Figure 3.9) which receives items coming from the two input conveyors and according to the priority of order decides which pallet should go as first to the output conveyor.

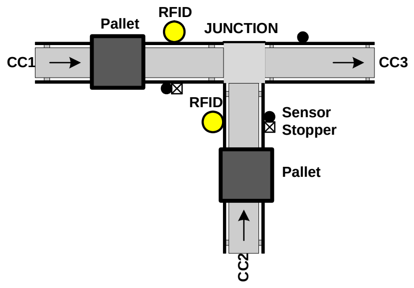


Figure 3.9: Junction unit

The major transportation tasks are: to deliver a part from the storage, to carry an unfinished subassembly between the machines, and to remove the finished product from the system. The intersection units as well as the index station units additionally have sensors and stoppers for the regulation of the pallet movements. The main agent classes of the pallet transfer system are presented in Figure 3.10.

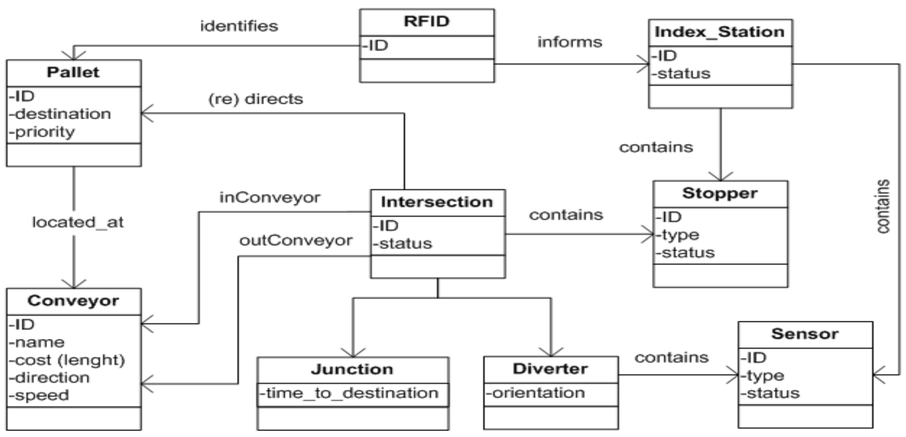


Figure 3.10: The main agent classes of the pallet transfer system

Since agents in our architecture do not have a global overview, decisions can be short-sighted. To avoid this and to facilitate the communication and coordination between the SA and the MAs, we decided to adopt a service agent provided within the used JADE platform [JAD08], the ***Directory Facilitator Agent*** (DF) [She99]. All resources contained in the platform should have their services registered at the DF. In case that SA or MA need to communicate or negotiate with other agents wherever in the system, these agents should contact the DF to find (the addresses of) the related agents and after a positive answer communicate directly with those agents. This approach significantly reduces the number of messages in system, since messages are only sent to a limited number of agents instead of being broadcasted to all agents.

3.4.2 Introduction of new Orders

Orders, placed by customers, are received by the contact agent. They contain information about the type of ordered products, quantities, and sometimes delivery dates. The CA splits these orders into product orders that are related to particular product type and for each of them creates one responsible OA and SA, respectively. The OA is skilled to produce the assembly plan for each product that can be produced by the factory plant and contains all the knowledge to produce the product, namely the product structure and the logistic process plan (Figure 3.11). The OA uses the ontology-based product model to extract the required parts and material for the production and contacts the storage agent to check their availability. If the remaining stock quantity is too short, it generates supplying order, forwards it to the CA and waits until the reception of required material or parts is confirmed. In the case that enough material or product parts are on stock, the OA uses the knowledge about the product, its parts as well as relationships between them to identify operations to be done on these parts and to generate work orders to be completed to finish this product. It forwards generated work orders to the SA with the request to start the negotiation with machine agents about supply. If the delivery date is provided, the OA uses this information to set the production priority. The SA sequences the related tasks from work orders, selects the first task necessary for work order finishing and specifies the related operation. In order to get list of available

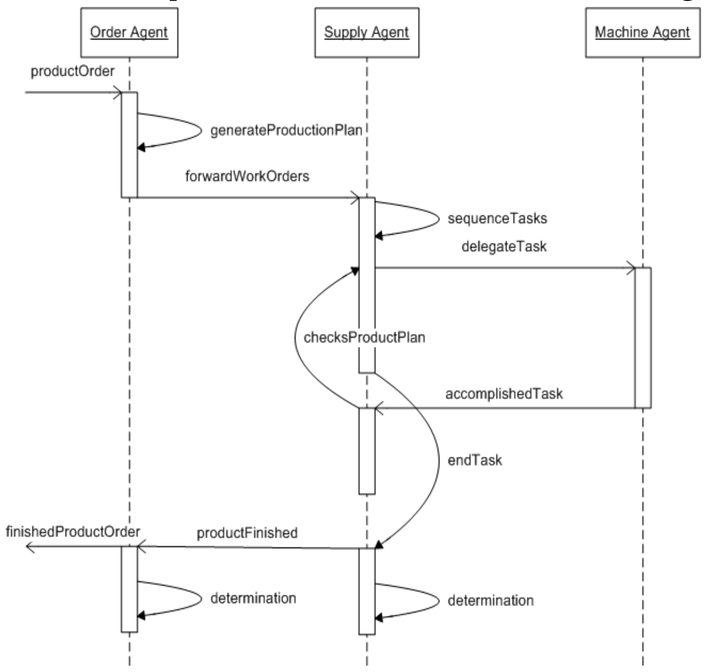
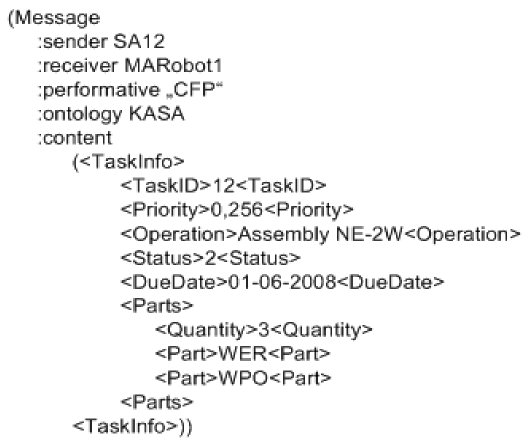


Figure 3.11: Introduction of a new Product Orders

resources for this operation, it sends the REQUEST message to the DF Agent with a specification of the required services. Because the DF agent covers only its platform, in order to be able to offer reliable information, this agent builds a network of DFs – the so called DF Federation [Tha04, Mer08c]. It detects services offered on other platforms. After the DF Agent sent the INFORM message to the SA about available resources, this agent starts the CNP and sends the CFP to all resources, respectively MAs, in the list (Figure 3.12).



```
(Message
   :sender SA12
   :receiver MARobot1
   :performative „CFP“
   :ontology KASA
   :content
      (<TaskInfo>
            <TaskID>12<TaskID>
            <Priority>0,256<Priority>
            <Operation>Assembly NE-2W<Operation>
            <Status>2<Status>
            <DueDate>01-06-2008<DueDate>
            <Parts>
               <Quantity>3<Quantity>
               <Part>WER<Part>
               <Part>WPO<Part>
            <Parts>
      <TaskInfo>))
```

Figure 3.12: CFP Message

Each MA calculates its agenda (the work that is already assigned to its machine) and sends the total processing time for all assigned tasks ($Tagenda_{total}$) with a PROPOSE message back. The processing time of the task from CFP is also included in the proposal since one machine can perform the same task quicker than other machine (e.g. different machining speed). The SA chooses the best proposal (resource with the lowest workload) and sends the ACCEPT_PROPOSAL message to the related MA with more precise specification of the operation. Since the destination of the task is known now, the SA can start to supply the resource with required parts/material. In the case that there are some parts to pick up, the first destination will be the storage. In order to find the best suitable pallet for transportation, the SA has to contact all available pallet agents (PA). The SA will again contact the DF Agent with the REQUEST message and "Transport" as a service specification. All pallets which are currently performing transport operations have their status set as *busy* and due to a specific implemented behavior are immediately deregistered from the DF Agent service lists. This state will be changed as soon as the pallets become free again. However, after the SA has received the list with available pallets, it starts the CNP and contacts all PAs. Having always accurate information about the pallet position, the PA calculates its distance to the destination and sends this as an answer [Mer08]. The nearest pallet to the destination will be chosen and the storage informed about its arrival. After it supplies the pallet with goods, the storage agent notifies the SA about supply and the SA forwards the information about the approaching pallet to the related MA. This information is very important because the operation with the highest priority will be loaded only if the parts are already available or on their way. On the way to the destination, the pallet will be routed based on shortest path algorithm as described in [Mer08]. When the first task was finished, the SA starts the execution of all others using the same procedure. At the end of the production order execution, the SA passes the relevant information to the order agent. The OA forwards this information to the CA, which waits until all product orders related to this customer order were accomplished to inform the customer about its shipment.

3.4.3 The Ontology

„Finding ways to reduce time and cost for getting high-quality and accurate knowledge through advanced information systems is a hot issue in current distributed information environments“[Qiu05]. The ontologies and embedded semantics can be used to formalize the knowledge representation and to achieve overall „understanding“. Individual agents, having their own objectives, knowledge and skills, may have different world models. In order to ensure correct understanding of the exchanged information agents must have the same presentation of the environment, or at least that part of the shared environment about which they are exchanging information with each other. We are using the ontology to provide semantic understanding among software agents. The ontology-based assembly model was presented in [Kim06] and serves as a formal, explicit specification of the assembly design so that it makes assembly knowledge both machine-interpretable and shareable at the same time. We use the same concept (Figure 3.13) to link product designs, assembly planning processes

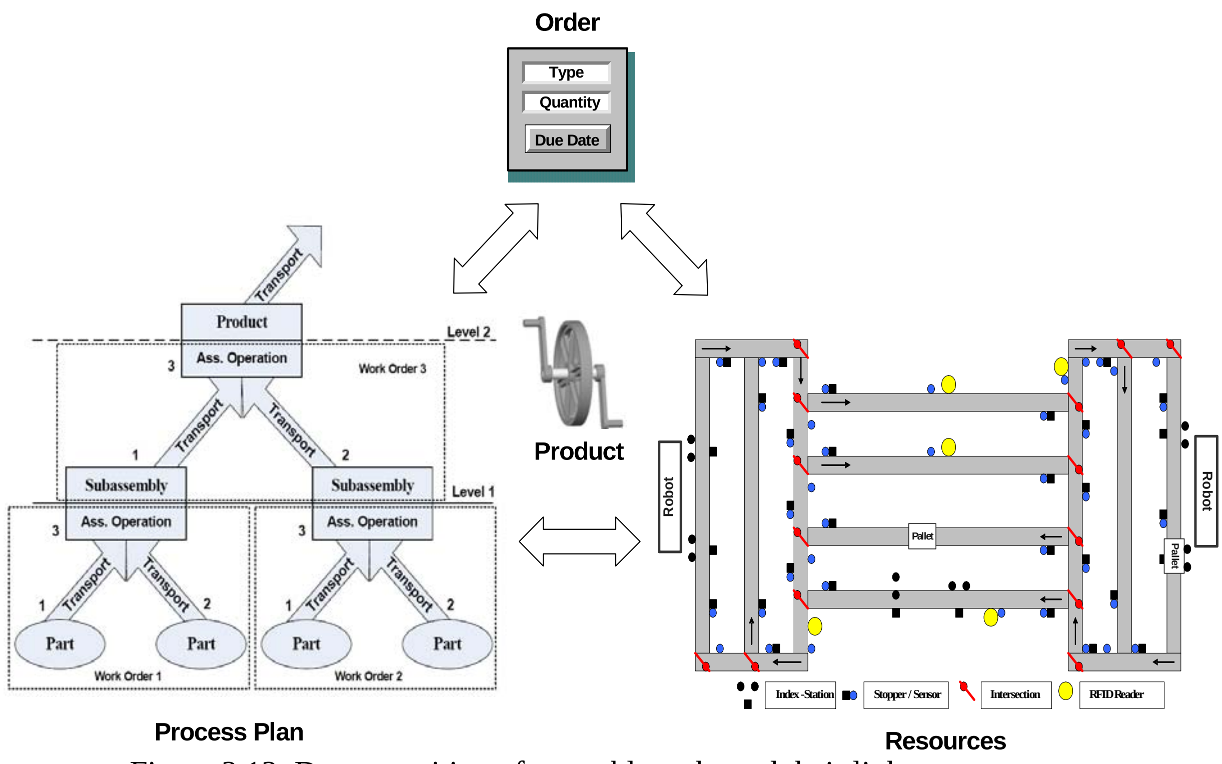


Figure 3.13: Decomposition of assembly tasks and their link to resources

and required assembly equipment together. The ontology based product model is used to extract the production/assembly operations from the product design and link particular tasks, which have to be performed for the production/assembly of a product, to particular resources. Each task consists of a series of actions that can be executed by one or more resources. These resources as well as their control are represented in the ontology. The connection to the product order is made through the type of ordered product, quantity, which defines the number of parts that have to be available to start the assembly process, and due date that defines the priority of the order.

As a basis for our ontology we take the "Machine Shop Information Model" [McL05] developed at the National Institute of Standards and Technology (NIST) as a part of efforts that support the development of standard data interfaces. This information model is intended to be used for representing and exchanging machine shop data, initially between manufacturing execution, scheduling, and simulation systems. The "Machine Shop Information Model" does not include any assembly specific concepts and constraints. Our ontology is also heavily influenced by the OZONE ontology [Smi97], the Enterprise Ontology [Usc98] and the ADACOR ontology [Lei04]. Our proposal is built on three basic layers: product, activity and resources (Figure 3.14). There is some correlation between our ontology and the FABMAS ontology [Mön05] as well as the MASON ontology [Lem06], which are both also based on three layers.

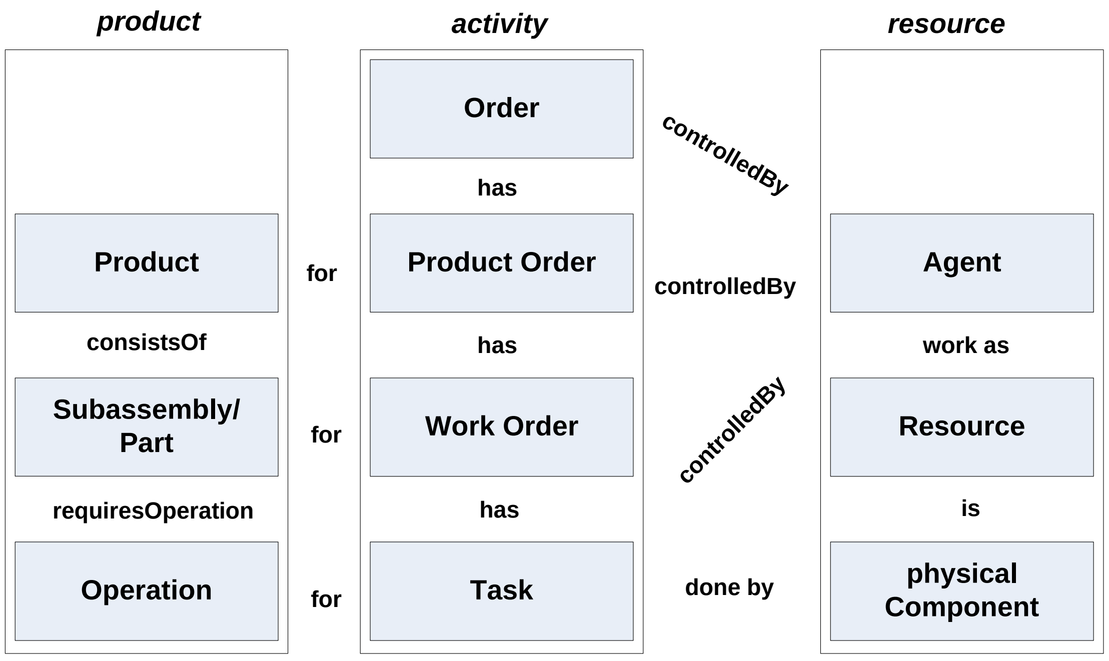


Figure 3.14: Three Ontology layers

A ***product*** is presented as a hierarchy of subassemblies and parts together with all their properties and relationship between them. Parts are defined as components, described by a set of attributes, properties, constraints and relations to other parts. A subassembly is a non-empty subset of parts that either has only one component or is such that every part has at least one surface contact with another part in the subset [Rab93]. The relationship between parts within a subassembly defines operations that have to be done to connect these parts and represents how these subassemblies should be put together to complete the product. An operation is defined as a discrete set of actions which leads to a certain change of state in or on the part. In our framework, we distinguish two types of operations: manufacturing/assembly and transportation operation.

An ***activity*** is the basic action, which specifies how the product state is changed. An activity describes how the product is going to be produced and how its production relates to all other entities in the production environment. Our concept describes the order activity as a bunch of product orders that split up further into sets of work orders, each work order being described as a list of tasks. In order to simplify the operation generation of the process plan, we adopt a top-down function model of the product [Zhu00]. Each product type is described through its own process plan. A *process and logistic plan* in the assembly tree specify the sequence of manufacturing or assembly operations which have to be performed in order to make a product. Each operation could be performed by different resources and consequently can be accompanied by related transport operations, making all together *work orders* (steps). Assembly phases are presented as levels in the process plan, which means that parts from the lower level will be assembled first (Figure 3.15).

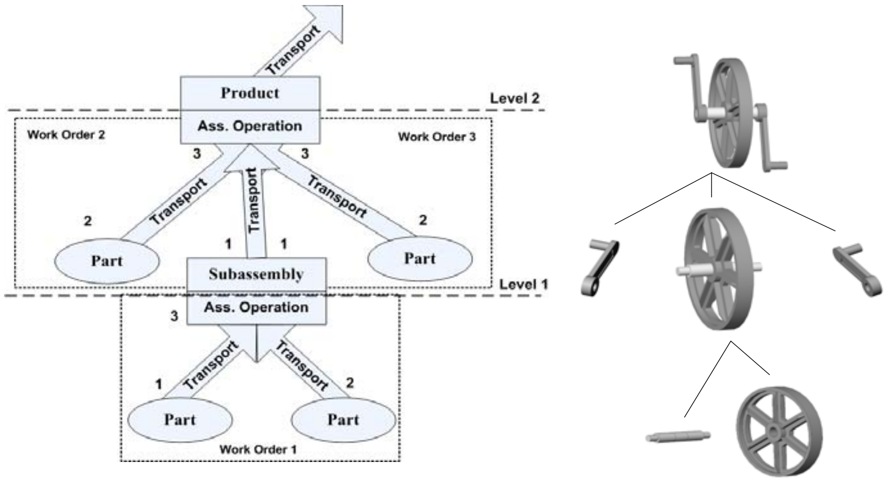


Figure 3.15: Logistic plan in the assembly tree

There is also a hierarchy within the same level and between work orders, where is distinguished which part/subassembly should be transported and assembled before others. We assume that each assembly operation is performed by a different resource which means that between two assembly operations it will be necessary to transport a part/subassembly to another destination. In the case that two assembly operations were performed by the same machine, one after another, the SA that manages the assembly will delete this transport operation.

A ***resource*** is a physical component able to perform a certain action. However, since this component embodies agent as its control part, we consider also the agent concept as integral resource element.

The agent interaction with the ontology in the background ensures that when an agent extracts relevant information from a message it understands the meaning of the terms in the message and the way this terms are combined in the statement. The presented concept distribution and ontological representation of a production process improves the way components communicate and exchange information in the manufacturing environment. Our ontology covers the environment structure, characteristics, states and components interrelationships enabling the related agents to interpret their environment, reason about it and make right decisions (Figure 3.16). Especially the structure of the *activities* layer, which has to serve as a link between the product layer (what is going to be produced) and resource layer (who is going to do this), fulfils its purpose by integrating the production planning and scheduling as well. The ontology-based concept of the product production/assembly described with *Steps* ensures the exact decomposition of the product orders to related work orders and further associated tasks and their correct indexing. This is particularly supported with the integrated planning relationships *needsPredecessor* (Step) and *isFollowedBy* (Step) that enables the SA to reason when and why to start particular task allocations, which are know as scheduling activities.

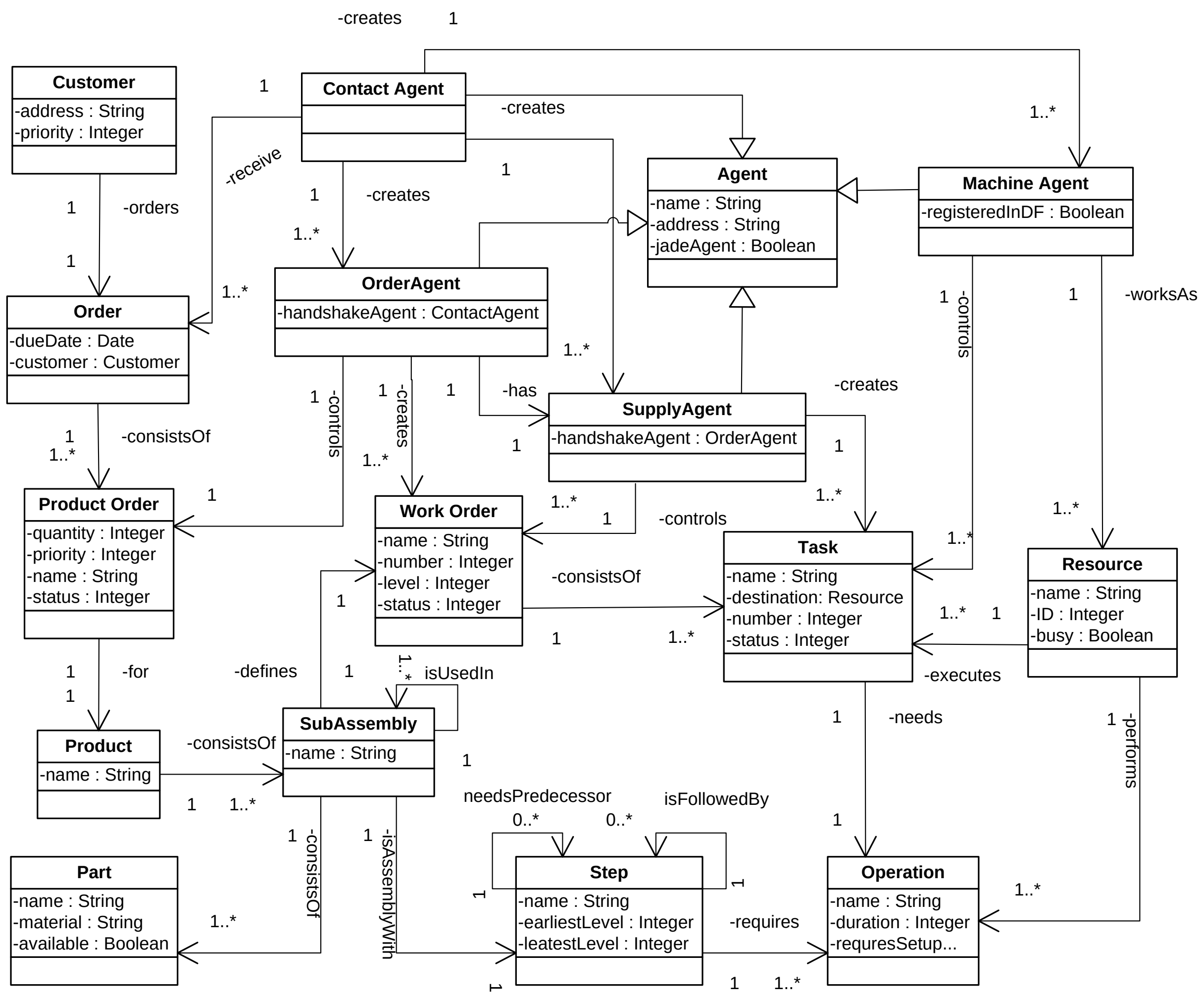


Figure 3.16: Assembly System Ontology

The important advantage of the introduced ontology-based approach is the achievement of the preconditions for easy assembly and disassembly of the product. Our knowledge-based system does not need to be told, how a problem has to be resolved (i.e., which and when particular tasks have to be done), but the concept and the goal is described instead. The system decides on its own how to achieve the goal.

3.5 Agent architecture

Our architecture consists of agents which are acting based on their knowledge. Each agent is an autonomous entity and has its domain of application. The agents have to observe their environment in order to possess its accurate representation (e.g. by sensors…). Having only a partial view of their surroundings, the agents are forced to communicate and cooperate with other agents in order to overcome the lack of a global perspective. Both ways, sensor and messages, of the agent's dealings with its surroundings have to be considered when designing its architecture. Since the physical decomposition approach presents combination of functional (agent part) and a resource part, it will be discussed in the rest of the chapter as representative case.

Considering the strict requirements of the manufacturing environment for real time action, our architecture splits the control of a manufacturing resource to the low level control (LLC) and high level control (HLC) (Figure 3.17). These levels are organized in a hierarchical way, where the HLC is represented as an agent, which uses the world model representation to coordinate the actions of a manufacturing resource and to delegate specific tasks to the LLC controlling indirectly its execution. On the other hand, the LLC governs the particular actions of the underlined physical system by collecting and processing the information from sensors. Based on the result, it performs some "reflex" action or informs the HLC about an event.

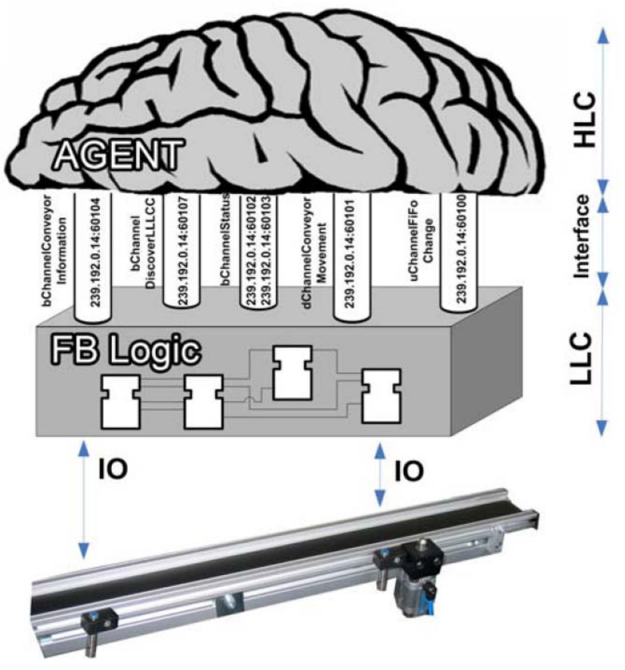


Figure 3.17: Control architecture of a component (conveyor)

3.5.1 The HLC Architecture

The HLC level, which is also known as the decision-making level or agent, is capable to govern the interaction with the external world and to understand, depending on the decomposition approach, different types of information that can be observed in the external material world. Here the decisions are made when and how, but also which actions are going to be performed at particular time. This level is able to reason about its own processes, characteristics, capabilities and goals. Based on the success or failure in achieving these goals, the decision about future steps will also be made on this level. The agent needs to have knowledge about its domain of application, about strategies, which can be used to achieve a specific goal, and sometimes knowledge about the (other) agents involved in the system. We consider an agent as an entity that has a set of protocols, which govern the operations of the manufacturing entity, a knowledge base, an inference mechanism and an explicit model of the problem to solve (Figure 3.18). The knowledge base stores the domain-specific knowledge. It is composed of an ontology and a set of problem solving rules. The crucial element in the decision component is the rule-based system, which applies declarative knowledge, expressed in a set of rules, to regulate the agent's behavior. The knowledge base together with an inference mechanism is the "brain" of the agent. The "inference" presents a process that indicates the generation of new conclusions from existing knowledge. The commonest basis for inferencing is rule considered with if-then statements applied to the knowledge base. If all defined conditions for a particular rule on the left hand side of the rule are satisfied the rule's actions on the right hand side will be executed. The inference engine recognizes and matches which rules in the knowledge base can be satisfied by provided facts in the working memory. If one or several conditions are not met, no rule will fire and the agent will wait for external events. In our case, rules are used to specify a set of agent actions to be performed for a given situation.

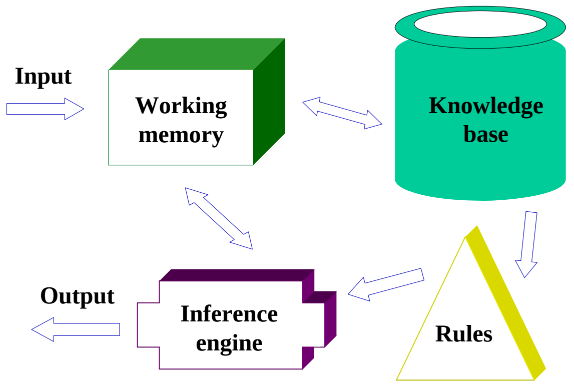


Figure 3.18: The decision architecture

The knowledge base is constantly updated with new facts causing the execution of new rules. This will again add new facts to the knowledge base and set conditions for firing new rules. A simplified rule where the order agent checks if enough parts are available for a product order is presented in Figure 3.19. Once a product order is issued, the order agent counts the parts needed for production or assembly and if enough parts are available it reserves these parts for this particular product order. Otherwise it sends a message to the CA requesting more parts.

The advantage of this type of knowledge-based system is to have a simple and very comprehensive way to represent the reasoning capability of an agent. This especially due to the high level of abstraction while expressing a particular agent's behavior by rules. Moreover, rules are usually faster and cheaper to program then imperative code concerning their relative independence from other code as well as easier understanding and maintenance.

```
(defrule OA_reserve_Parts
     (object (is-a Order_Agent)
       (OBJECT ?oa)
       (responsible_for ?po))
     (object (is-a Product_Order)
       (OBJECT ?po)
       (name ?npo)
       (quantity ?q)
       (product ?p))
     (object (is-a Product)
       (OBJECT ?p))
   (object (is-a Contact_Agent)
       (OBJECT ?ca))
   =>
   (bind $?pl (slot-get ?p parts))
   (foreach ?part $?pl
     (bind ?count 0)
     (bind ?x (count-query-results find_Part ?part))
           (if (< ?x ?q) then
           (bind ?result (run-query* find_Part ?part))
           (while (and (?result next) (< ?count ?q))
           (bind ?count (+ ?count 1))
                 (bind ?objp (?result getObject part))
           (slot-set ?objp reserved TRUE))
        else
           (send_msg
               (make-instance of Message
                 (performative "REQUEST")
                 (sender ?oa)
                 (receiver ?ca)
                 (content "Need_Parts"))))))
```

Figure 3.19: A simplified rule

The important part of the agent is its interaction model used for knowledge exchange, problem or conflict resolving, as well as negotiation, cooperation and coordination with other agents. This model provides the basic communicating system and supports the negotiating process. Autonomous agents cooperate by sending massages and using concepts from domain ontology. Our agents communicate by sending ACL massages designed especially to describe and facilitate agent communications and to make it possible to exchange information or knowledge between the heterogeneous agents [Fou03]. Having an architecture that involves autonomous agents, we do not need to predefine the interaction for the whole system, since it emerges from the actions and behaviors of the constituent agents.

3.5.2 The Low Level Control

The Low Level Control layer is built on top of the mechatronic components [Sun05] (Figure 3.17). Basic mechatronic components are compositions of the physical hardware and the interface provided to the control hardware (e.g. pneumatic or electrical connectors) to actuate or sense. Mechatronic components can also be built as a composition of multiple mechatronic components. For example the mechatronic component "Conveyor" consists of several components:

- the conveyor,
- the motor (offering electrical connectors to supply the motor and move the belt),
- a power relay, allowing to switch the direction of the conveyor belt movement and offering electrical connectors, that are apt to be directly connected to low current micro controller outputs, and
- inductive sensors to sense the presence of pallets [Heg08].

The low level control is established as a distributed control application based on IEC 61499. This layer includes the control hardware, the interfaces to the mechatronic components, the IEC 61499 runtime environment, and the control application [Heg08]. IEC 61499 is a new standard family of the IEC for Industrial Process Measurement and Control Systems (IPMCS). The standard IEC 61499 defines several models—the application model, the system model, the device model, the resource model, and the Function Block (FB) model—that allow the control engineer to develop distributed control applications in a graphical manner. The base model of IEC 61499 is the FB (Figure 3.20). An FB is a software component that is self contained and provides its functionality through a defined interface. A trigger on one of the event inputs starts the execution of an FB. During the execution of the FB the input data will be processed, output data will be generated (depending on the functionality of the FB), and/or output events will be triggered. IEC 61499

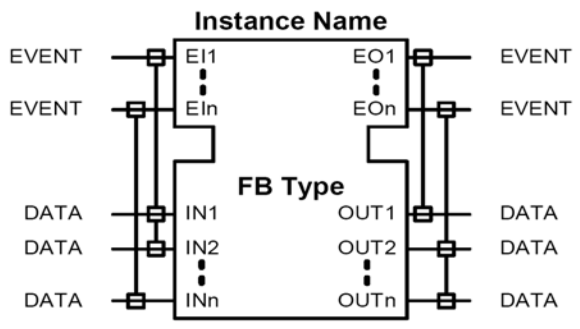


Figure 3.20: Graphical representation of the interface of an IEC 61499 Function Block [ZoiDiss07]

defines three different FB types:the ***basic FB*** that contain as main element a state machine that controls the internal execution on an input event arrival***;*** the ***composite FB*** that serve as container for FBs and may contain a whole set of FBs and their event connections and data connections; and the ***service interface FB*** that provides an FB interface with functionalities which are beyond the means of IEC 61499 (e.g. access to the control device's hardware, like the I/O interface or the communication interface). As said in the state of the art section, IEC 61499 has several advantages that make this standard suitable for the basic control software architecture for reconfigurable lower level industrial control. First of all, it bases on IEC 61131-3 which it makes easier to switch to the new technology. Furthermore important to notice are its modularity, its support for distribution, the event-triggered execution model, and a basic reconfiguration support as the key features for using this architecture as basis in this work [ZoiDiss07].

3.5.3 Communication Interface between Agents and the Low Level Control

An important aspect of our concept is the existence of a run-time communication interface allowing to transfer the information from the RT (real time)-control subsystem (i.e. data from sensors, diagnostic subsystems, etc.) to the agents and, vice versa, to propagate the control actions decided by the agents to the RT-control subsystem and thus to the physical actuators. The requirements on the interface between these LLC and HLC layers were specified by Hegny et al. [Heg08]. The interface shall be independent from the specific protocols, procedures, media, and software platforms. Furthermore a loose coupling of the layers (the LLC is subjacent to the HLC layer) shall guarantee that the interface does not disproportionately burden the LLC, which is capable to keep real-time constraints. The event-driven execution of IEC 61499 and the asynchronous communication paradigm of software agents fit well. Therefore, mechanisms of IEC 61499 can be used easily for the interface. Service Interface Function Blocks (SIFB) encapsulate the access to functionalities provided by the controllers (e.g. timers, network access, and inputs and outputs). However, to implement such a communication interface few aspects of communicative commonality have to be considered as well: the means of communication should be standardized and accepted by all involved participants as well as the communicative acts have to be also implemented in a form that enables overall understanding. In our architecture HLC-LLC communication is done by sending a message with specific content through predefined channels which are known by both sides [Mer09]. Channels encapsulate the transmission paths between the LLC and HLC (Figure 3.17). These can be classified into upstream, downstream, and bidirectional channels, depending on the direction of communication. Upstream means that the message will be sent to the high level, downstream on the other hand means that messages are sent to the low level using a specific channel. Besides, the channels are specified with their own IDs. Few different implementations of the channels are possible:

- Network-based: both layers communicate over the network, using widely spread network protocols like UDP over Ethernet or TCP/IP [Heg08].
- Shared memory: when both HLC and LLC are placed on the same controller, local inter-process communication such as shared memory can be applied [Lop07].

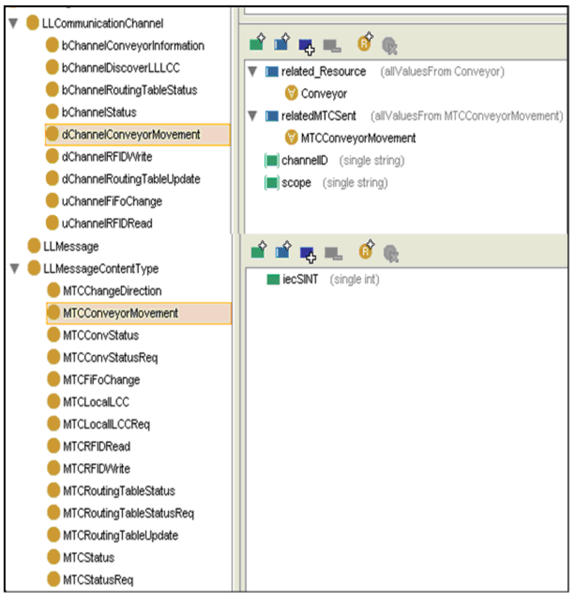


Figure 3.21: Ontological representation of a LLC message

On the LLC side the implementation of the interface is done with SIFBs. These define all aspects of the communication. On the HLC side, to make the interface usable for agents an appropriate description in the ontology is required. Here are again followed the aspects that separate the semantics of message transport from message type. In terms of IEC 61499, a message is defined as a set of data transferred via a SIFB triggered by an event. Consequently, message types in the ontology have to represent a fix-ordered list of IEC 61499 data types defining the structure and encoding of the message [Heg08]. A representation of these datatypes is included in the ontology as well as the MessageContent concept, and its subconcepts. These concepts specify an ordered set of IEC 61499 datatypes, which are used to transmit information between the layers. The unique name of a message type gives information about the purpose of the message (e.g. MTCConveyorMovement – content related to the conveyor directions regulation) and can therefore be checked for correct structure by the transport mechanisms (Figure 3.21). Channels encapsulate the actual transport mechanism. They cover issues such as addressing, protocols or content validation. The Class Channel provides a generic channel representation; it consists of scope (global or local), a unique ID, an assigned related resource type, and an assigned message type for each applicable direction. A generic interface for the HLC-LLC interface, which encapsulates the chosen implementation, allows an easy exchange of the transport mechanism between HLC and LLC entities, which represent the same mechatronic component, and to transmit and receive information (i.e. messages) on the existing, previously established communication channels. Status updates and commands are messages that are necessary to keep HLC and LLC synchronized.

Changing a single aspect of the interface is possible without influencing the other aspects. For example the transmission channel can be replaced without the need to adapt the message content (e.g. replacing a multicast-message based network transmission with a shared memory implementation).

3.6 Implementation

In order to validate our approach, we implemented the multi-agent architecture presented above. The overall system has been built on top of the Java Agent Development Environment (JADE) framework [JAD08]. The JADE architecture enables agent communication through message exchange based on agent communication language (ACL) [Fou03]. We have used Protégé-2000 [Stanf07] as an integrated software tool to develop the knowledge base. The reasoning is implemented using the Jess expert system shell [Sandi07]. JessTab [Eri02] is used as a plug-in for Protégé that allows us to use Jess and Protégé together. The expanded description of the implementation as well as of the mentioned tools will be given in the Chapter 6.

3.7 Summary

In this chapter, we presented the KASA multi agent architecture with agent classes corresponding to the defined layered structure of manufacturing system and mapping related system goals. Our MAS is composed of distributed heterogeneous units/agents, where each agent has its own objectives, knowledge and skills, has ability to easy absorb the permanent changes in manufacturing organization and environment. The manufacturing system agility is improved using ontology to enable knowledge interoperations between agents and to ensure the same presentation of the shared environment about which they are exchanging information with each other. Moreover, we used the ontology to integrate product designs, assembly planning processes and required assembly equipment.

Taking into consideration the real world conditions, where particular actions have to be performed in a real-time, our architecture divides the control of a manufacturing resource to the low level control (LLC) and high level control (HLC). The HLC makes long term decision, has overview over the agent's knowledge, goals and skills and coordinates manufacturing resource actions with global manufacturing environment. The LLC based on IEC 61499 manages actions of subjacent physical resource, acquiring and processing the information from sensors at the same time. Such agent structure enables high flexibility, modularity and reusability of hardware and software components.

The next chapter will elucidate the dynamic workflow scheduling and performance of our architecture under dynamic conditions.

4. Dynamic Scheduling in the KASA

"It is circumstance and proper timing that give an action its character and make it either good or bad."
Agesilaus II (444-360 BC)

4.1 Introduction to Dynamic Scheduling

Manufacturing systems are getting every day more complex and dynamic. Assembly systems, which are integral part of these systems, are also faced with these trends being forced to produce customized products in a short time at low price and under dynamic conditions. These assembly systems should also be able to handle failure events, e.g., if a resource becomes unavailable or overloaded. In such circumstances, appropriately selected and executed scheduling is a key feature for maximization of the system output. Being able to optimize a schedule of tasks considering capacity and time constraints is of primary importance for achieving these objectives [Bab05]. Due to the significance of process scheduling, extensive research work has been reported [Pin02]. However, the traditional centralized control approach, due its rigidity and centralized structure, suffers from the lack of flexibility and reconfiguration abilities especially if unexpected events occur, e.g., resources become unavailable, variations in job processing times, or sudden changes in task priorities. Moreover, the central unit could be also seen as a bottleneck, which may limit the capacity of the shop, and as a single point of failure, which can bring down the entire shop [Oue07]. The multi-agent system (MAS) approach, based on decentralized control architecture, has been suggested as an alternative to centralized control and scheduling [Bus04]. The main MAS characteristic related to the scheduling is that within this architecture each agent is responsible for scheduling of its own tasks as well as for supporting global system scheduling through coordination and interaction with the other agents. The global scheduling emerges from the local agent's scheduling. The application of an agent-based approach in the scheduling domain tends a) to simplify the design and implementation of the scheduling system, b) to make the system more robust, and c) to reduce computational time and consequently the time needed to obtain a reasonable solution [Bab05, Jen03]. We employ MAS in combination with different dispatching rules to study dynamic scheduling strategies with particular attention on parallel machine scheduling strategies. In parallel machine scheduling, there are n tasks that have to be allocated respecting a set of constraints (capacity, due date, processing time, etc.) to m machines, each of which is running at its own speed. Our MAS assembly system is modeled as closed queuing transfer network with redundant paths, where each machine station connected to at least one transfer path (conveyor) can perform a particular operation. The order of tasks, which have to be accomplished to finish the product, is fixed and each product as well as the related tasks can have different due dates.

4.2 Task Scheduling

The performance of the production system is affected with the choice of optimal strategies. Especially, if there exist recourses having scientifically bigger workload than other. In this case the machine with the maximum workload is the bottleneck. Balancing the workload between available resources will help to avoid such preventable limits and improve system output. Balancing the workload between parallel machines, considering the current system state, has been recognized as a potential way to flexibly schedule appropriate operations. Particularly in complex environment, this approach can maximize the overall system throughput, minimizing the work in process, flow time, and makespan [Raj04]. Moreover, it can increase resource utilization and hence improve productivity; and at the same time can help to avoid bottlenecks and to schedule unavailable resources.

However, the parallel machine scheduling problem has two separate aspects: resource allocation and task sequencing. Resource allocation concerns the assignment of tasks to appropriate machines that can handle such tasks, while task sequencing prioritizes the tasks assigned to a machine. Our approach is structured according to these aspects: (i) resource allocation that is based on the negotiation between agents and (ii) task sequencing based on workflow balancing and dispatching rules for machine agents. Significant research results have been reported in the field of parallel machine scheduling [Che90, Mok01]. Agents in combination with dispatching rules have been tested for dynamic scheduling in [Yin07, Raj99, Won06]. Various agent architectures for dynamic scheduling in flexible manufacturing systems have been reported [She02].

4.2.1 Task allocation

Many optimal and approximation algorithms for solving different types of scheduling problems have been developed and tested [Bla01]. In the distributed approach, the tasks allocation and execution is a result of the coordinated activities of many agents. Moreover, additional constrains/parameters (capacity, due date, processing time, etc.), which are specifying the properties that must not be violated, have been used in order to specify the width of a valid scheduling solution. Each agent schedules its own actions according to its current knowledge about the environment. To stay coordinated, agents need to synchronize their activities using inter-agent communication. As mentioned in previous chapter, we are using CNP to assign the task to the machine with the lowest workload. Our approach introduces the SA, whose main responsibilities are: (i) to send a call for proposals on tasks allocation to machine agents, (ii) to receive proposals for a particular task from the machine agents ($T_{agenda_{total}}$), (iii) to compare the proposals, and (iv) to finally allocate the tasks. The main objective for applying the negotiation strategy is to balance the workload among the resources evenly.

4.2.2 Task sequencing

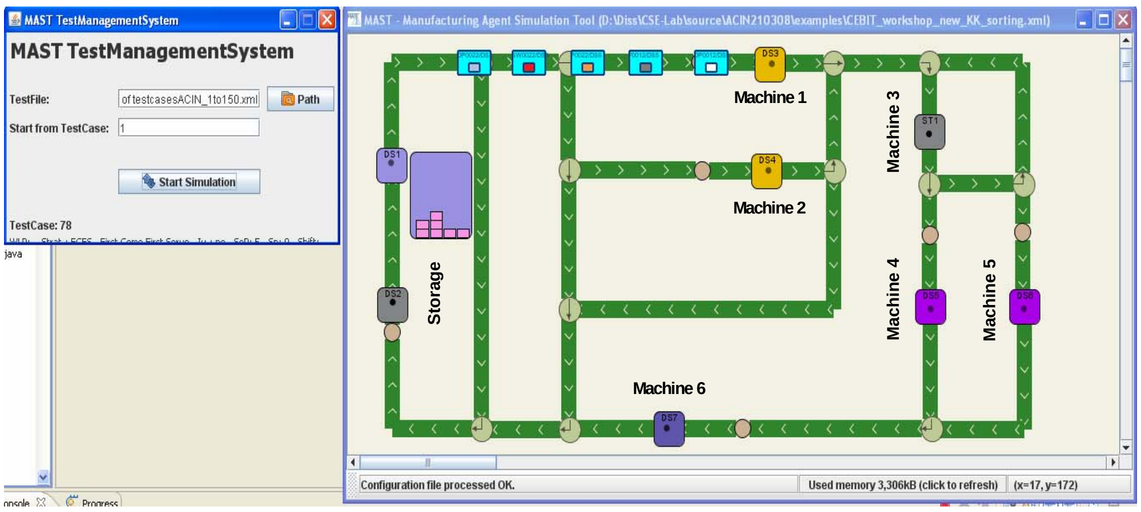
The dispatching rules have been used by MAs for sequencing the tasks allocated to their machines. Dispatching rules are extensively applied in manufacturing practice due to their simplicity, effectiveness, and their nature of an on-line algorithm [Hol97, Pes01]. Their usage for the local scheduling optimization of production lots with regard to various indicators, e.g., due date, machine utilization, has been a long-established research field [Cha02, Raj99]. Moreover, the structure of dispatching rules corresponds to rule-based behavior of our agents and is easy to implement. MA is using the dispatching rules after the current task is completed for selection of a next suitable one from a set of tasks that are awaiting for being processed by a machine. At this time, the related subassemlies/parts are already on the way to the machine, being immediately transported after ACCEPT PROPOSAL from bidding phase (if there are pallets available). In order to select the best suited dispatching rule for implementation in our system, we created four scenarios in which MAs alternately tested one of the following rules:

1. *First Come, First Served (FCFS):* The first allocated task is executed first.
2. *Critical Ratio (CR)* expressed as: $$CR = \frac{Rt}{Pt} \qquad (4.1)$$

   where R_t is the remaining time from the current time to the due date of a product and P_t is the sum of processing times of all remaining tasks that make up that particular product. A lower CR indicates that the task has a tighter due date or longer processing time. A task with a lower CR is given higher priority than tasks with higher CR.
3. *Earliest Due Date (EDD)* gives the highest priority to the task with the earliest due date. The task inherits the due date from the related product.
4. *Shortest Processing Time (SP):* The task with the shortest processing time is sequenced first.

4.2.3 Simulation approach

We have used the Manufacturing Agent Simulation Tool (MAST), which is able to provide agent-based simulation support for our empirical study [Vrb08], to simulate various scenarios. The Test Management System (TMS) is used for automatically run predefined sets of test cases described in XML files [Mer08d]. These tools are easy to couple with our system architecture, since both systems use the JADE framework and almost identical MAS architecture [Mer08e]. The characteristics of the used architecture and the implemented production workflow are presented in previous chapter. The simulated Test bed layout is presented in Figure 4.1.


MAST TestManagementSystem

TestFile: oftestcasesACIN_1to150.xml

Start from TestCase: 1

Start Simulation

TestCase: 78

MAST - Manufacturing Agent Simulation Tool (D:\Diss\CSE-Lab\source\ACIN210308\examples\CEBIT_workshop_new_KK_sorting.xml)

Configuration file processed OK.

Used memory 3,306kB (click to refresh)

(x=17,y=172)

Figure 4.1: Simulated MAST Layout and the Test Management System.

A total of 600 test cases have been generated with the TMS. Each test case consists of a workflow scheduling strategy (FCFS, CR, EDD, SP), the number of pallets to use (5, 10, 15 and 20) and a workload of 40 orders. An order consists of a product type to be built and a randomly generated due date (defined in seconds) for the product. For simplicity, three pre-defined product types were used – simple, medium and complex – that differ in the number of machine operations and raw materials/semi-products needed to assemble the final product. So called product trees depicting both required raw materials (ovals) and assembly operations (rectangles) for these three product types are shown in Figure 4.2. The shift time for a test case was set to 10 minutes (600 seconds). This value was chosen in order to ensure that 40 randomly generated orders could not easily be finished without a proper workflow scheduling strategy. This implies that the overall time needed for producing all workload items is larger than the set shift time, resulting in less items being produced than actually ordered. Otherwise, if all ordered products would haven been produced, we would not have been able to measure the effectiveness of the chosen simulation parameters by using the number of finished products as a reliable parameter.

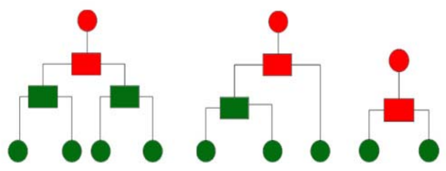


Figure 4.2: Product types

4.2.4 Simulation Results

Two kinds of measurements were carried out: a) the measurement of the effect of the number of pallets used together with the workflow scheduling strategy selected on the *number of finished products*, and b) the measurement of the impact of the selected workflow scheduling strategy on the *machine utilization* rates. The following section describes the results of the first type of measurements in detail (the second type results can be found in [Mer08d]).

The number of finished products is defined as the sum of all products which have been produced within a given shift time. The number of finished products measures the effectiveness of the workflow scheduling strategy parameterization, e.g., number of finished products in relation to the number of pallets used. In Figure 4.3 we show the number of pallets as significant predictor for the average number of products finished within a shift. Note that above a certain number of pallets the effectiveness increase due to faster transportation declines, for one strategy (FCFS) the number of finished products is even lower with a higher number of pallets.

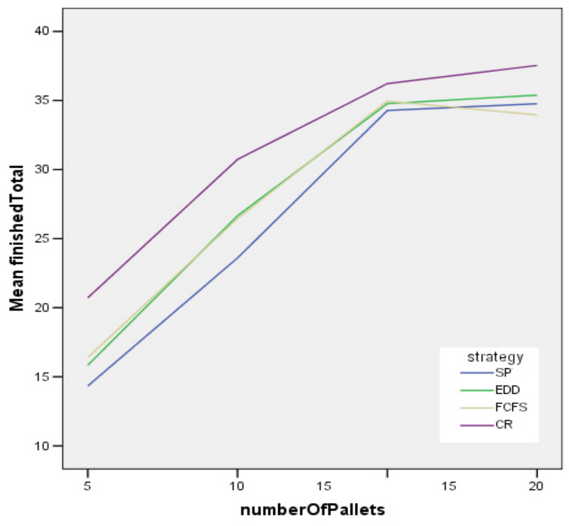


Figure 4.3: Number of finished products within a shift.

	Number of Finished Products			
	SP		FCFS	
Pallets	**Mean**	**STD**	**Mean**	**STD**
5	**15,83**	**1,21**	**16,40**	**1,14**
10	**26,66**	**5,95**	**26,46**	**5,59**
15	**34,77**	**10,27**	**34,96**	**6,53**
20	**36,03**	**7,06**	**33,95**	**8,91**
	EDD		**CR**	
Pallets	**Mean**	**STD**	**Mean**	**STD**
5	**14,33**	**1,33**	**20,60**	**1,67**
10	**23,22**	**5,70**	**30,73**	**4,31**
15	**32,58**	**7,59**	**36,22**	**4,75**
20	**34,76**	**5,92**	**37,54**	**3,95**

Table 4.1: Comparison of Number of Pallets and Number of Finished Products.

In the experiments we used different numbers of pallets (5, 10, 15 or 20) combined with workloads of 40 products randomly selected out of 3 different product types (simple, medium, complex). Table shows that the combination of Critical Ratio (CR) as workflow scheduling strategy and different number of pallets provides the highest number of finished products in average and could be considered as the best strategy. The CR, Shortest Processing Time (SP), and Earliest Due Date (EDD) strategies have shown that by adding more pallets into the workshop the number of finished products within a shift will increase. Just to give examples: by increasing the number of pallets from 5 to 10 the average number of finished products will increase by 68 % in SP, 62% in EDD, 61% in FCFS and 49% in CR. However, we can also observe while analyzing FCFS that by increasing the number of pallets from 15 to 20 the average of finished products will drop by 2% - therefore we conclude that FCFS has better usage for lower number of pallets compared to other strategies such as EDD and SP.

The reason is that the SP, EDD and CR strategies always utilize the maximal number of pallets to deliver parts from the storage to the machines for assembling in parallel. In contrast, the First Come First Served (FCSF) strategy uses only a limited number of pallets, since the SA will not send further orders to the production line, until the first product is assembled.

4.3 Task Scheduling Considering Transportation Times and Conveyor Failures

Traditional calculations for workflow scheduling strategies focus only on machine service duration; however, in some contexts the variation of transport time is a significant scheduling factor. It is possible that the distance to the selected resource and transportation time needed could significantly influence the efficiency of the overall system [Byr97], especially when the transportation times between machines are considerably longer than the machine processing times. Lee and Chen as well as Hurink and Knust studied machine-scheduling problems considering transportation time and capacity [Byr97, Lee01]. Subsequently, Lee et al. considered disruption management and rescheduling [Hur01]. However, applying this approach in a centralized control system (with the associated difficulties in handling a large complex system) leads to an exponential growth in the number of possible scheduling solutions [Lee06, Hah94]. Moreover, since the manufacturing processes regularly change their states and settings, which may make previously optimal plans suboptimal; thus, we see the need for a more flexible approach able to provide reliable solutions in “near-real time”. We have investigated how balancing control policies in combination with transportation time calculation influence the overall system performance in production under different operating conditions.

4.3.1 Task allocation and sequencing

The task allocation procedure starts when the SA sends an announcement message to all MAs which offer the required machine function. The MAs answer with a bid message containing the estimated processing time of the machine function plus the estimated time needed for the transportation to the machine. The SA then chooses a MA with the minimal sum of machine function and transportation time and allocates the current task to this MA and the represented machine respectively by sending a bid confirmation message to the particular MA. On the other hand, the machine agents are, as described in previous section, using different dispatching rules (*FCFS, EDD, CR* and *SP*) to select the next job in the waiting queue. Additionally, we extended the CR and SP rules by adding the transportation time to machine to the calculation of the processing time, which is calculated in order to select the next task to be scheduled.

1. *Critical Ratio + Transportation Time (CRT):* Defined as quotient of the sum of processing times of all remaining tasks *(Pt)* for a product and transportation time (T_t) to the machine for this particular task with the remaining time from the current time to the due date of the product (*Rt*). The task with the lowest CRT is selected.

$$CRT = \frac{Rt}{Pt + T_t} \qquad (4.2)$$

2. *Shortest Processing + Transportation Time (SPT):* Defined as the ratio of the sum of the duration of the next task *($T_{task_{total}}$)* and the transportation time (T_t) to the machine for this particular task with the total processing time of the product *($T_{Product_{total}}$)*. The task with the lowest SPT is selected.

$$T_{Pnoduct_{total}} = \sum_{i=1}^{n} T_{task_{total}}(i) \qquad (4.3)$$

$$SPT = \frac{T_{task_{tota\ddot{o}}} + T_t}{T_{product_{total}}} \qquad (4.4)$$

The performances of selected dispatching rules have been empirically evaluated and compared. The MAST is used again to build scalable and flexible production systems with an underlying transportation system (Figure 4.4). The TMS has been used to measure system performance under stable conditions and when facing unexpected events (e.g., failures of the transport system - conveyors), which influence the variation of transport durations.

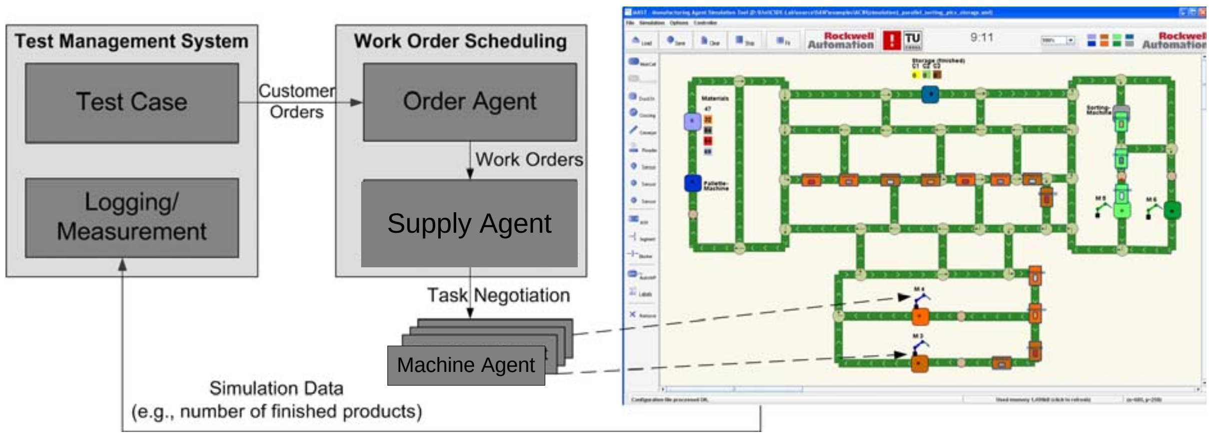


Figure 4.4: Overview System Architecture

4.3.2 Simulation approach

A total of 1,085 test cases were generated from the scheduling strategies as input to the TMS. Each test case consists of a scheduling strategy (FCFS, CR, EDD, SP; CRT, SPT), the number of pallets to use (10, 15 and 20), failure specifications and a workload of 25 orders. An order consists of a product type to be built (Figure 4.2) and a randomly generated due date for the product. The shift time for a test case was set to 600 seconds to ensure that 25 randomly selected orders could not easily be finished in the study context without a proper workflow scheduling strategy (see Section 4.2.3).

The failure specification consists of the identifier of the affected resource to fail, the start and end points in time of the occurrence of the failure. We classified the risk of a failing conveyor (according to the position and the importance of the conveyor for the overall system) for all conveyors in the workshop to 5 failure classes (see Table 4.2). For effective comparison of the robustness of workflow scheduling strategies regarding their exposure to failures in the transportation system, failures with the same specification were used for all workflow scheduling strategies.

Failure Class	Failure Impact
C0	No failure
C1	Failures of redundant conveyors which cause almost no detours
C2	Failures of redundant conveyors which cause long detours
C3	Failures of conveyors resulting in the unreachability of a single redundant machine
C4	Failures of conveyors resulting in the unreachability of multiple machines

Table 4.2: Failure Classes and Risk Analysis

4.3.3 Simulation Results

In this section we present the results of the data analysis, which signify the change in production effectiveness due to a) the application of transport time consideration into the scheduling approaches and b) introduction of several classes of transportation failures. Figure 4.5 shows that adding more pallets increased the number of finished products within a shift which holds true for all strategies. In Figure 4.5, when using 10 pallets, SPT offers the best overall production performance (*Mean: 15.6* and *STD: 1.8*), however when increasing the number of pallets to 15 and 20 respectively, CRT outperforms the others strategies.

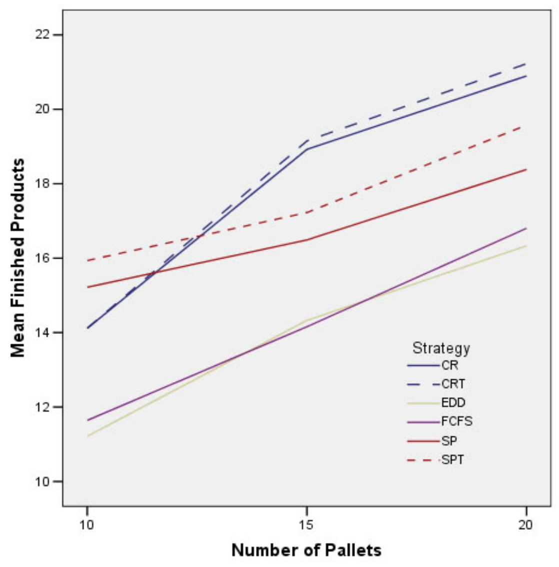


Figure 4.5: Production effectiveness without failures for 6 work scheduling strategies

Number of Pallets		Number of Finished Products					
		CR		**CRT**		**EDD**	
		Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**
	10	**14,11**	**1,07**	**14,11**	**1,45**	**11,21**	**1,29**
	15	**18,92**	**2,81**	**19,15**	**3,37**	**14,32**	**2,28**
	20	**20,89**	**3,38**	**21,22**	**3,70**	**16,33**	**3,12**
		FCFS		**SP**		**SPT**	
		Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**
	10	**11,64**	**2,30**	**15,21**	**1,60**	**15,65**	**1,75**
	15	**14,15**	**2,83**	**16,49**	**1,98**	**16,67**	**2,42**
	20	**16,80**	**3,23**	**18,38**	**2,00**	**18,88**	**2,31**

Table 4.3: Average number of finished products for 6 work scheduling strategies (no failures)

In Table 4.3 we can see that by increasing the number of pallets in the workshop from 10 to 15, the average number of finished products increases too, depending on the scheduling strategy: CR (33%), EDD (26%), FCFS (23%), and SP (4%). Additionally, we analyze the impact of including the transportation times in the calculation of the CR and SP dispatching rules. Figure 4.5 outlines both extensions offer slightly better results as CRT improved the performance of CR by average of 3 %, while SPT improved the performance of SP by 5%.

In the second scenario, we analyze the results of introducing transportation failures into the simulation. We ran 210 test cases consisting of 10 test cases for each combination of strategy with class of failure. Therefore we can analyze the performance of each strategy by given the higher possibility to reach their maximum performances. Our data analysis suggests that introducing failures of a higher failure class statistically will likely reduce the number of finished products (Table 4.4). Figure 4.6 shows that without any failure all strategies offer their best performances compare to scenarios with failure, CRT and CR provide better performance in finishing the products as ordered while coping with failures compared to the other strategies. In Table 4.4, adding the first failure class reduced the average number of finished product by: CR (5%), CRT (7.4%), EDD (11%), FCFS (6%), SP (1%) and SPT (8.8%).

We have conducted an empirical study on the impact of scheduling strategies (dispatching rules), number of pallets available, and transport system failures on system performance, measured the number of finished products in a shift. For this empirical study we ran extensive simulation tests for data collection and statistical data analysis. The system performance, measured with and without inclusion of transportation times, has shown some improvements when transportation times were included in the calculations. In order to

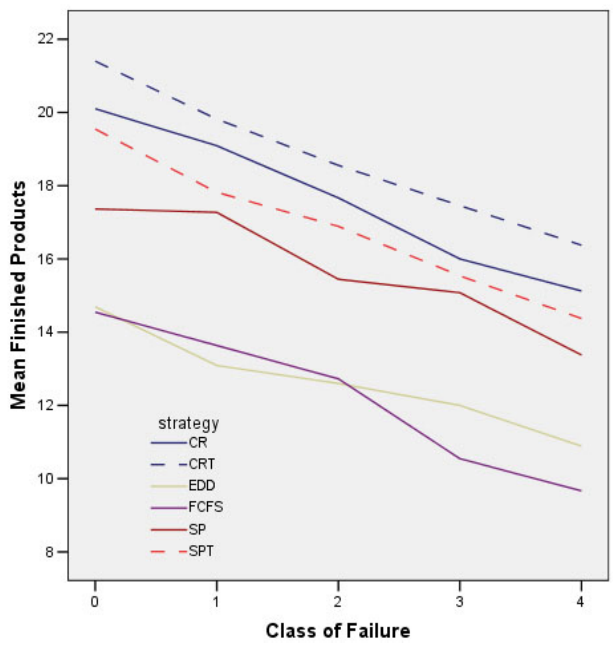


Figure 4.6: Production effectiveness with failures for 6 work scheduling strategies and 20 pallets

		Number of Finished Products					
		CR		**CRT**		**EDD**	
		Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**
Failure Classes	**0**	**20.10**	**2.60**	**21.40**	**3.30**	**14.69**	**3.75**
	1	**19.09**	**1.44**	**19.81**	**1.72**	**13.09**	**2.42**
	2	**17.66**	**1.41**	**18.55**	**2.69**	**12.60**	**2.50**
	3	**16.00**	**1.26**	**17.45**	**1.75**	**12.00**	**2.09**
	4	**15.12**	**.835**	**16.37**	**1.18**	**10.88**	**1.83**
		FCFS		**SP**		**SPT**	
		Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**
	0	**14.54**	**2.29**	**17.36**	**2.29**	**19.54**	**2.01**
	1	**13.63**	**3.58**	**17.27**	**2.32**	**17.81**	**2.22**
	2	**12.72**	**2.14**	**15.44**	**1.50**	**16.88**	**1.36**
	3	**10.54**	**2.06**	**15.07**	**1.11**	**15.53**	**1.39**
	4	**9.66**	**1.37**	**13.30**	**1.06**	**14.37**	**1.06**

Table 4.4: Average number of finished products for 6 work scheduling strategies (with transport failures) using 20 pallets

investigate the ability of agents to manage dynamic environment conditions (such as machine failures) in the production automation domain applying dynamic dispatching rules and diverse failure handling mechanisms, we have done extensive tests by measuring system robustness and systematically comparing the overall system performance (e.g., number of finished products). This research will be presented in the remaining part of the chapter.

4.4 Re-Scheduling Using Machine Failure Handling Policies

Besides the rapidly changing market environment and customer requirements, current manufacturing systems also have to face dynamic conditions during the production process. Machine breakdowns or disturbances, task priority changes, integration of new resources, order cancellations, unequal machine utilization rates, and product quality problems are some of many exceptions that can influence the system performance and typically occur unpredictably. If the system is able to handle a particular fault situation and continues to operate without a significant loss of functionality, it is called fault tolerant [Hag96]. Fault tolerant systems should exhibit proportional degradation of service (e.g., throughput) depending on the class of problem that occurs. There is a range of failure handling policies to respond to exceptions and thus to improve the system tolerance. Traditional centralized hierarchical manufacturing systems, due to their rigid structure and lack of flexibility, suffer from weak failure tolerance, i.e., they are not able to handle such events effectively and efficiently, meaning that they either stop working or produce less products. Moreover, in these traditional systems all possible combinations of exceptions have to be predicted at the design time, otherwise their occurrence in real time can lead to scheduling errors and significant downtime related losses. Nevertheless, in complex systems the number of combinations grows exponentially which makes system re-scheduling and modification very expensive or time consuming [Tic06].

The application of the multi-agent systems based on decentralized control architecture has been suggested as a promising approach for overcoming these difficulties [Bus04]. Nevertheless, a blind application of the MAS approach to increase fault tolerance of a particular system can lead to the opposite result [Tic06]. It is necessary to empirically investigate MAS performance under dynamic conditions, when agents use a range of promising behaviors and apply diverse handling policies in order to cope with system exceptions.

Considering the scheduling of production resources as a one of the key features of production control, in this section we examine the influence of re-scheduling on production effectiveness. Failure re-scheduling policies specify the overall tactics that define *when* and *how* the system has to cope with failure events. Several handling policies for task re-allocation carried out in case of extra-ordinary events (e.g., machine breakdowns) are explored in this study. In the section 4.2, we evaluated a range of workflow scheduling strategies based on multi-agent negotiation, where each resource agent performs local scheduling using dispatching rules for sequencing the tasks allocated to their machines. Considering their advantages, the usage of dispatching rules for sequencing rescheduled tasks after a specific handling policy is applied can reduce effects of a particular exception and improve system performance [Vie03]. Kutanoglu and Sabuncuoglu [Kut01] used this approach to study four reactive scheduling policies – no rerouting, queue rerouting, arrival rerouting, and all rerouting – developed for rerouting the jobs to alternative machines when their primary machine fails. Bastos et al. [Bas05] presented a multi-agent architecture capable to support dynamic resource allocation planning in production environments. This architecture also can manage disturbances in the production system in real time by applying two strategies – replacement and re-scheduling. A market-based theory coordinates agent behaviors. Wu et al. [Wu07] presented an algorithm for automatic sequential resource (re)allocation among a group of agents in complex environments with limited shared resources and with uncertainties. Vieira et al. [Vie03] extensively studied the effects of re-scheduling policies on the performance of a manufacturing system. They concluded that the use of different model types, such as a mathematical model of dynamic and stochastic manufacturing systems, queuing network model or discrete event simulation model, can give useful information to analysts. However, the mathematical model does not explicitly represent the production control policies that will be actually used to control the system. Furthermore, additional research is required to compare the performance of a manufacturing systems under diverse dynamic conditions to explain the advantages and disadvantages of re-scheduling in different problem settings.

Moreover, we simulate the real-life scenarios to test system performance in a dynamic environment. In this paper, we evaluate the failure tolerance of several re-scheduling policies from literature, where agents negotiate to coordinate their actions and apply dispatching rules for local scheduling. We also investigate how specific production conditions such as different levels of machine efficiency as well as duration of machine failures influence the performance of a handling policy. In order to strengthen the external validity of our research results, we use the real-world pallet transfer system at the Institute for Automation and Control, Vienna University of Technology, as a reference model for our MAS architecture.

In the rest of this section, we evaluate the failure tolerance of several re-scheduling policies from literature, where agents negotiate to coordinate their actions and apply dispatching rules for local scheduling. We also investigate how specific production conditions such as different levels of machine efficiency as well as duration of machine failures influence the performance of a handling policy. In order to strengthen external validity of our research results, we simulate the real-life scenarios to test performances of our MAS architecture in a dynamic environment.

4.4.1 Re-scheduling Policies

The way how manufacturing systems treat the exceptional events can significantly influence their performance. Using predefined schedules, such systems are doing well if every-thing is going well. However, it is of vital importance to define their reaction on unexpected events. A re-scheduling policy specifies what event triggers re-scheduling and what method will be applied for re-scheduling. Moreover, it also specifies the method applied for revision of the existing schedules [Vie03]. Three policies related to the re-scheduling initiation events have been presented in literature [Sab00, Oue07] – periodic, event-driven, and hybrid. A periodic policy is periodically initiated with a defined time period during which all available system information is collected and then used for deriving the re-scheduling setting. However, the effectiveness of this policy depends on the optimally adjusted length of the period, which might be hard to effectively anticipate. Moreover, this policy is not agile enough because critical events, which require prompt reaction, are not processed immediately, but wait at the end of the re-scheduling period. This is not the case for event-driven re-scheduling triggered immediately when the specific event (e.g., job arrival, machine failure) occurs. However, in a large system, where the number of such events happening simultaneously can be enormous, the application of this policy can lead to continuous rescheduling and thus to lower stability and performance. A hybrid re-scheduling policy can be seen as combination of previous two approaches as the system re-schedules periodically as well as when specific, user-defined events occur, synchronizing policy occurrence avoiding their overlapping (e.g. to have periodic and event-driven rescheduling at the same time).

Having a smaller manufacturing system as a test case, we decided to apply an event-driven re-scheduling policy, considering machine failures as events that trigger re-scheduling. We implemented and tested four agent-based schedule repair methods corresponding to methods presented in [Kut01]:

1. *Right-shift scheduling (RS):* when a machine breaks down this method postpones the job being currently processed as well as all other jobs that are waiting in the machine's queue until the machine is repaired. During the repair time period the machine agent (MA) that is in the charge of this machine still responds to calls for bids from supply agents (SA), offering its free capacity in "after-repair" period.
2. *Agenda rerouting (AR):* after the machine fails all jobs from the machine's queue are rerouted to alternative machines. In contrast to previous case, the time that the jobs loose by waiting in the machine's queue for its repair can be saved. However, also in this case the MA bids on its services during the negotiations with the SA about new jobs.
3. *New jobs rerouting (NR):* in this case the MA keeps all jobs in the machine queue while refuses to bid on new arriving jobs. This policy tends to avoid the additional load to failed machine by not accepting new job arrivals and to prevent system stress through the machine queue jobs rerouting.
4. *Complete rerouting (CR) policy:* combines the AR and NR methods. The machine's MA addresses the SA's of jobs that are scheduled to the failed machine to reroute the jobs to alternative machines as well as does not participate in subsequent negotiations with SAs about new jobs.

Due to its simplicity, the RS policy is mostly applied by current manufacturing systems. Thus we consider it as a reference policy in our study. The AR and NR could be referred to as partial scheduling policies while the CR represents a regeneration policy. As mentioned in the introduction section we are additionally applying dispatching rules to handle dynamics of manufacturing environment. These rules are used by MAs, which supervise functioning of machines for optimal sequencing of allocated jobs. The important advantage of dispatching rules is the ability to select the highest priority job from the machine's queue considering all reliable up-to-date information at the time of selection. We are using the Critical Ratio rule to prioritize jobs, because (as presented in section 4.2.4) it showed the best performance in comparison to others dispatching rules.

4.4.2 Research Issues

In the context of the empirical study, we define system performance, equivalent with the production effectiveness function *E*, as the average number of finished products within a given shift. As argument we consider following parameters: level of workload, number of pallets, re-scheduling policy, and type of failure. We define the following research hypotheses to be validated by the experiments:

RI1: Influence of number of pallets on system performance *E*. Let us assume that a higher number of pallets will result in a higher overall system output. *IF n* and *m* are the number of pallets introduced, and *E* is the production effectiveness function, *THEN* we define the following null hypothesis:

$$\boldsymbol{H_0\text{-}1}: \{\exists \mathrm{m} > \mathrm{n} \mid E(\mathrm{m}) \leq E(\mathrm{n})\} \quad \text{(eq. 4.5)}$$

RI2: Impact of re-scheduling policy on system performance *E*. Let us assume that the use of different re-scheduling policies will result in a diversity of resulting system outputs, because some re-scheduling strategies include the machine agents representing broken machines into the negotiation process, while others do not. Furthermore, the different handling of already queued or future awarded jobs to broken machines could also have a significant impact on the overall system performance. *IF α* represents the AR re-scheduling policy, *β* represents the NR re-scheduling policy and *γ* represents the CR re-scheduling policy, *THEN* we can define the following null hypothesis:

$$\boldsymbol{H_0\text{-}2}: \{\forall \alpha, \beta, \gamma \mid E(\gamma) \leq \min(E(\alpha) \vee E(\beta))\} \quad \text{(eq. 4.6)}$$

RI3: Impact of failure type on system performance *E*. Let us assume that the duration of machine's unavailability (two different periods for machine failures and machine disturbances) has a direct influence on the overall system performance. *IF f* is a machine disturbance (**f**aster recovery) and *s* is a machine failure (**s**lower recovery), *THEN* we propose the following null hypothesis:

$$\boldsymbol{H_0\text{-}3}: \{\forall \mathrm{f}, \mathrm{s} \mid E(\mathrm{f}) \leq E(\mathrm{s})\} \quad \text{(eq. 4.7)}$$

RI4: Impact of workload level on system performance *E*. Let us assume that the number of orders directly influences the relative system performance. *IF l* represents a lower number of received orders and *h* represents a higher number, *THEN* we identify the following null hypothesis:

$$\boldsymbol{H_0\text{-}4}: \{\exists h > l \mid \frac{E(l)}{l} \leq \frac{E(h)}{h}\} \quad \text{(eq. 4.8)}$$

4.4.3 Simulation Approach

This section describes the details of the simulation experiment carried out in MAST-TMS environment. A total of sixty four evaluation scenarios have been tested. In each test case, the Critical Ratio dispatching rule has been applied and different values of following parameters set: the re-scheduling policy (RS, AR, NR and CR), the number of pallets providing transportation (10, 15, 20, or 25), the machine failure recovery time (f - faster recovery or s - slower recovery) and the level of workload (l – low: 2,880 or h – high: 5,760 number of orders). An order consists of a product type to be produced and randomly generated due date (in seconds). We used three pre-defined product types as shown before in Figure 4.2. Machine operation times and transportation times were considered as fixed for all evaluation scenarios. The shift time for an evaluation scenario was set to one full day (24 hours) to ensure that depending on the used workload a set of randomly selected orders could not easily be finished in the study context (see Section 4.2.3).

The machine failure specification consists of the machine identifier and start/end time points of the failure. We classified the risk of failing a machine using two different failure classes: machine disturbances (f – faster recovery), which can be re-paired in approximately 0.1% of the overall shift time, and machine failures (s – slower recovery), which requires a longer repair time, in our case about 10% of the overall shift time. For effective comparison of the robustness of the re-scheduling policies, failures with the same specifications were used for all evaluation scenarios. We used a workshop layout consisting of four machines, each capable of performing between 2 and 4 different machine operations, presented before in Figure 4.4. The sixty four generated evaluation scenarios were split down into four batches, each containing sixteen test cases and particular re-scheduling policy applied. These batches were run in parallel on four high-performance mainframe servers, taking it approximately eight hours to finish a single batch. After all tests had been finished, the resulting data were collected from the mainframe servers and analyzed using various statistic tests.

4.4.4 Experimental Results and Discussion

We explore the impact of additional factors such as failures, disturbances, overloads, etc. on the system output. Especially, we judge the manufacturing system ability to absorb a machine failure using predefined failure tolerance policies. As important factors we consider different duration of machines down-time, diverse levels of production workload as well as various number of transportation units (pallets).

A) Number of finished products considering all factors

Similarly to Kutanoglu and Sabuncuoglu in [Kut95], but applying the decentralized multi-agent control and using CNP for workload balancing, we reached the same conclusion – the superiority of CR (complete rerouting) policy over the other policies. As presented in

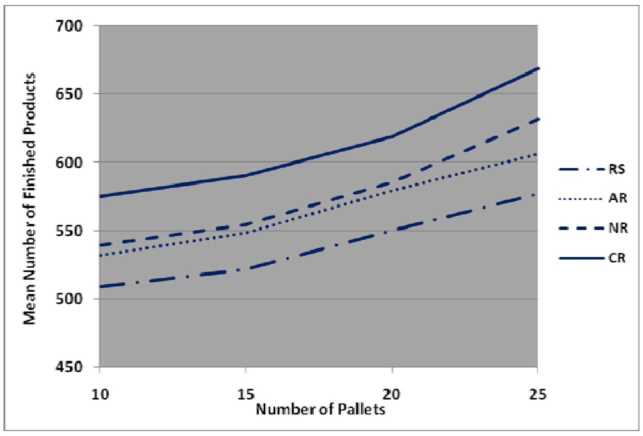


Figure 4.7: Mean of finished products of a shift by strategy

Strategy	Number of Pallets							
	10		15		20		25	
	Number of finished Products							
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
RS	509	64	522	68	551	71	577	75
AR	532	71	549	71	580	75	606	78
NR	539	69	555	72	586	77	632	58
CR	575	74	591	77	619	86	669	66

Table 4:5: Finished products by number of pallets and re-scheduling policy.

Figure 4.7 as well as in Table 4.5 showing the mean numbers of finished products depending on number of palettes, the CR policy outperforms RS (right-shift scheduling) policy approximately by 12% to 16%. This is an expectable result because the immediate exclusion of the failed machine from production and full re-scheduling of tasks in its queue to other available machines significantly compensates the failure state when no action would be taken in case of RS policy (the tasks in machine’s queue have to wait for machine repair). This is of course true only if suitable alternative resources are available.

Notable is a good performance of the NR (new jobs rerouting) policy, which proves that the failed machine should be urgently excluded from further scheduling until repaired. How-ever, due the fact that the NR policy keeps already awarded jobs in machine’s queue, its performance is approximately by 5.6% to 6.6% weaker than of the CR policy. Interesting fact is also the influence of number of pallets on production performance. The 150% increase of pallets results only in 15% gain in production output. This can be justified by increased number of traffic jam situations caused by a limited space for palettes in a conveyor-based transportation system.

B) Number of finished products focusing on failure duration

Extensive experiments have also been done to identify how the efficient failure recovery impacts the system performance. As mentioned earlier, in a general overview, the CR-(f - faster recovery or s - slower recovery) policy wins the race, but relation between the NR-(f,s) and AR-(f,s) policies deserve deeper study (consult Figure 4.8 and Table 4.6).

While the difference between the NR-f and AR-f policies, when short disturbances periodically occur, stays almost constantly around 1% in favour of the NR-f policy, this ratio notably grows when the failure duration prolongs (NR-s and AR-s curves respectively) and the number of pallets rises. The fact that the NR-s policy performs by 8% better than AR-s when using 25 pallets can by explained such as this policy is handi-capped just by those products which are already awarded to the machine while all others incoming after the failure are re-balanced to other available machines. On the other hand, the AR-s policy profited by re-scheduling of the queued jobs, but only to such a point where the number of palettes in the system caused traffic jams and consequently increased the number of unfinished products that were on their way to alternative machines instead of waiting in the machine's queue for repair. This finally resulted in weaker performance of the AR-s policy.

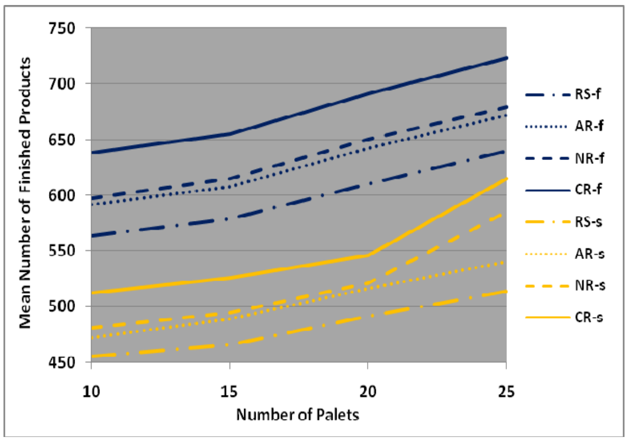


Figure 4.8: Influence of failure duration (f - faster recovery or s - slower recovery) .

Strategy	Number of Pallets							
	10		15		20		25	
	Number of finished Products							
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
RS-f	564	71	579	75	610	79	640	83
AR-f	592	79	608	79	642	83	672	86
NR-f	598	76	615	80	650	85	679	62
CR-f	638	82	655	85	692	96	724	71
RS-s	455	57	466	61	491	63	514	67
AR-s	472	63	489	63	517	67	541	70
NR-s	481	62	495	64	521	69	585	54
CR-s	512	66	527	69	547	76	615	61

Table 4,6: Finished products by defined failure duration.

C) Number of finished products focusing on workload

The introduction of two levels of workload (l – low: 2,880 or h – high: 5,760 orders respectively) has again confirmed the supremacy of the CR-(l, h) policy (see Figure 4.9 and Table 4.7).

It is interesting to note here that the system with higher workload, especially due to the overloading and traffic jam, has shown a 4.7% to 6% weaker performance. Nevertheless, even then the CR-h policy with a higher workload was able to outperform all remaining policies.

D) Empirical Results

Analyzing the empirical results, we derive the following implications for the impact factor analysis regarding the re-scheduling policies:

Influence of number of pallets on system performance *E*. The data analysis shows that an increase of the number of pallets will increase the overall system output. As shown in Table 1 the number of pallets has a significant impact on the overall system performance (in terms of number of finished products) with a p-value < 0.01. In the study context, an increase of the number of used pallets (i.e., from *n* to *m*) always increased the number of finished products. Therefore we can state that $\{\exists m > n \mid E(m) > E(n)\}$, and so we can reject null hypothesis ***H*₀-1**.

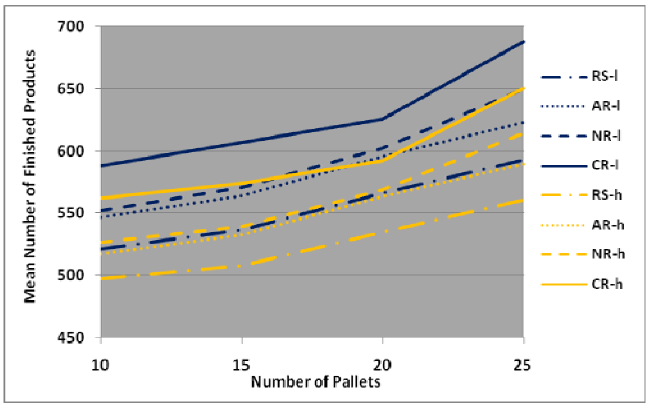


Figure 4.9: Influence of different levels of workload (l – low: 2,880 or h – high: 5,760 orders)

Strategy	Number of Pallets							
	10		15		20		25	
	Number of finished Products							
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
RS-l	522	66	537	70	566	73	593	77
AR-l	547	73	564	73	596	77	623	80
NR-l	552	71	571	74	602	79	650	60
CR-l	589	76	608	79	635	87	688	68
RS-h	497	62	508	66	535	69	561	73
AR-h	518	69	533	69	564	73	590	76
NR-h	527	67	539	70	569	75	615	56
CR-h	562	72	574	75	603	82	651	64

Table 4.7: Finished products by level of workload.

Impact of re-scheduling policy on system performance *E*. As shown in Figure 3, the Complete Rerouting (CR) policy outperforms all other rerouting policies (p-value < 0.01). Therefore, we conclude that the overall system performance of the CR policy (γ) is higher than the system performance of all other rerouting policies, namely the NR (β) and AR (α). We can state that $\{\forall \alpha, \beta, \gamma \mid E(\gamma) > \min(E(\alpha) \vee E(\beta))\}$, and hence we can reject null hypothesis ***H*₀-2**.

Impact of failure type on system performance *E*. Figure 4 shows that the failure recovery time has a direct impact on the resulting overall system performance. A faster failure recovery (f) will increase the production effectiveness, while slower failure recovery (s) decreases the production effectiveness with a p-value < 0.01. In the study context, a decrease of the failure recovery time always led to an increased number of finished products. Therefore we can state that $\{\forall f, s \mid E(f) > E(s)\}$ and reject null hypothesis ***H*₀-3**.

Impact of workload level on system performance *E*. As shown in Figure 5, the number of orders directly influences the relative system performance. As observed in the experiment, an increase of the number of orders (i.e., from *l* to *h*) will statistically lead to a decreased overall system performance. Therefore we can state that

$$\{\forall h > l \mid \frac{E(l)}{l} > \frac{E(h)}{h}\}$$

and reject null hypothesis ***H*₀-4**.

4.5 Summary

The simulation results indicate that the concept of multi-agent approach is well suited for building complex systems featuring a good performance. Moreover, due to its distributed nature it is possible to optimize the output of the system components consequently, resulting in an increase of the system efficiency. Our simulation results indicated that the critical ratio offers better results than the other tested rules. The system performance, measured with and without inclusion of transportation distance as well with and without failures, has shown some improvements when transportation times were included in the calculations. Comparing the different re-scheduling policies, we concluded that complete rerouting policy remarkable outperforms all other policies. Our experiments show that a higher number of pallets in fact always increased the number of finished products, but due to traffic jams with diminishing marginal gains. Further, longer failure durations have a certain impact on the overall system performance, since the number of products in the queue as well as the participation of a failed machine in the job allocation process seriously influences system performance. Finally, we found in the study context a limited size of workload to lead to higher system output, because of the lower system utilization and fewer occurrences of traffic jams.

This work can be leveraged in terms of implementing proactive re-scheduling policies and evaluating and comparing their performance to the performance of reactive re-scheduling policies. Additionally, more complex workshop layouts (more machines, more complex routes) could be used to compare the performance of the presented re-scheduling policies. Moreover, future work can leverage the results of this thesis by analyzing and evaluating combined impact of transport system failures and machine failures on the system efficiency.

5. Failure Tolerance in the KASA

"If everything seems to be going well, you have obviously overlooked something."
Murphy's Law

5.1 Introduction to Failure Tolerance

We noticed in the second chapter that serious concerns regarding the robustness and survivability, especially in complex and unpredictable environments, restrict wide adaptation of multi-agent systems in industry [Hel04]. Consisting of autonomous entities/agents that have to cooperate and coordinate their activities, the performance of multi-agent systems (MAS) could be significantly influenced with the failure of particular agent(s). While their modularity establishes MAS as good platforms for building fault tolerant systems on the one side, their non-deterministic behaviour makes it pretty hard to predict overall system output in fault situations on the other side [Sny04]. Especially in large-scale MAS, where the failure rate grows with the system complexity due to the number of deployed agents and the computation duration, the fault tolerance is a vital issue [Xua04]. These systems have to be able to absorb the failure (detecting, isolating and recovering from it), while the rest of the system proceeds with its regular functions.

Respecting its importance, a lot of work has been done and reported with regard to fault handling in multi-agent systems. The inability of an agent to perform a particular action, which was specified in its behaviour and triggered through an external event, can be defined as its failure. Several types of failures have been specified in literature: *crash failures* when a component stops producing output; *omission failures* when the faulty component can eventually resume its output production; *timing failures* occur when the output is produced outside the defined time slot; and *arbitrary failures* are related to arbitrary output values at arbitrary times [Fac06]. If the system is able to handle a particular fault situation and continues to operate without a significant loss of functionality, it is called *fault tolerant* [Hag96]. The failure tolerance has been mostly spread and analysed in two related domains: *failure detections* and *failure recovery*. The failure detection is an essential service that has to be provided in a system in order to achieve satisfactory failure tolerance. Several approaches that cope with the identification of failure states have been investigated [Dev00]. A typical scenario for failure detection is the periodical exchange of "I am alive" messages between individual hosts [Fac06]. The problem here is equivalent increase of the message flow that can cause an overload of the communication service and influence the system performance, if the system structure is not well organized. Bertier et al. presented a shared failure detection service split into two layers: the basic layer that adapts the heartbeat emission interval to the network conditions, and the adaptation layer that customizes the quality of service provided by the first one according to application needs [Ber03]. Hagg used another approach by applying external agents (sentinels), which observe the inter-agent communication. They can detect inconsistencies in the agents' behaviors by monitoring their activities as well as their internal state. Some items (beliefs) from the world model of agents, which are related to them, are directly copied to the world model of the sentinel. Other parts of the model can be built by monitoring the agent communication and by direct interaction (asking). Sentinels can also use timers to identify non-functional agents (or a faulty communication link) [Hag96]. A similar approach is reported by Snyder and Tomlinson who are using sentinel-based unreliable fault detectors for failure identification [Sny04]. The social diagnosis approach, where agents are comparing their own states with peers in particular team-mates to detect possible failures, is presented by Kaminka and Tambe [Kam98].

Failure recovery can be defined as the application of particular techniques for handling identified failures. A replication mechanism is often used for failure recovery [Fac06, Gue04]. It involves the replicated agents that have several copies of their behaviours and states on different locations. Once the original agent failed, its replica can overtake its place in the system. Mellouli presented a methodology based on this mechanism, which assists designers in the development process of fault-tolerant systems [Mel05]. However, the shortcoming here is a weak software redundancy since the identical software subsystems will fail in identical ways [Huh02]. In the previously presented sentinel approach the failure recovery is done by sentinel agents, which are choosing alternative problem solving methods to recover an agent from a failure state [Hag96]. Nevertheless, the architectural structure, where sentinels can fully inspect and influence other agents, does not satisfy the assumptions of openness and agent autonomy [Pla08]. On the other hand, Kumar and Cohen presented a fault tolerant brokered architecture, where the teamwork is used to recover a multi-agent system from broker failures [Kum00]. However, it seems that the system is more focused on the broker failure tolerance and less on those of individual agents, requiring extra computing for the management of brokerage layers [Kha05].

Having a multi-agent architecture, which is based on agents that encapsulate particular functions of manufacturing systems and only have software representation on one side as well as agents that supervise specific hardware components within manufacturing systems on the other side, we present a hybrid failure tolerance approach that combines the heartbeat mechanism for failure detection and the supervisor agent approach for system failure absorption and recovery. In the rest of this chapter, we will describe specifications and differences of failures related to the agent's representation type as well as the failure tolerance procedure linked to software or hardware agents respectively. At the end of the chapter we will present implemented system recovery scenario and discus achieved performances. Due to its simplicity, we will use the crash type of failures as case study.

5.2 Failure Types

Multi-agent systems can be seen as a community of autonomous, intelligent entities/agents, where each agent has to cooperate and coordinate its activities with other agents in order to achieve own and respectively the system goal as well. Being mostly applied in a dynamic environment where the permanent changes are the only constant, agent has to be able to identify these changes. An agent can receive information from the environment by sensing it (e.g. sensors) or through communication (e.g. messages) with other agents. In order to react appropriately on it, an agent has to be able to “understand” this information. Consequently, in the case of an exceptional event (e.g. failure) an agent has to be able to interpret it. Moreover, an internal representation (e.g. knowledge) is required as reference for agents to evaluate incoming events and to be able to distinguish the exceptional from ‘normal’ ones [Pla08].

Having its surrounding represented in an ontology, an knowledge-based agent is able to reason about the concept and appropriately respond on the received information. It is capable to discover its own or even failure from other agents if required. Moreover, its actions can be saved in the knowledge base, which copy could be placed and permanently updated on the “safe” location. In the case of the identified agent failure, the new agent with the same behaviours and copied knowledge from the old one can be relaunched in order to reach the predefined goals. Differences to the replica approach are that here are copied only the data and not an agent and that there is no divestiture on critical and non-critical agents since every agent has particular meaning to the system. We successfully tested this scenario, but since this was not the main focus of our approach it will be not discussed further.

As presented in the third chapter, our architecture is based on the two types of agents: physical and functional ones. Receptively, MAS based on different agents types result in diverse failure types and related failure tolerance procedures as well. Considering the definition of Hiller, a system failure is a result of a system state error, which again is caused by a specific fault [Hil98]. It is also important to mention that not every fault leads necessarily to a failure. Nevertheless, each particular failure, which can potentially influence the system in a bad way (i.e. by minimizing its performance), has to be detected and possibly absorbed. We will make a brief look into possible failure types of MAS, which highly depend on the nature of the application. In process automation the following three viewpoints could be identified: physical component failures, software entity failures, and system disturbances. Physical component failures include the complete breakdown of particular resources or their temporary failure (e.g. blockade of an intersection, overloaded conveyor, etc.). Software entity failures include all types of failures related to agents in MAS. Mellouli distinguished agent failures regarding the agent’s ability to communicate [Mel05]:

The communication with the agent is down but:

- The agent can perform all of its tasks,
- The agent can only perform some of its tasks,
- The agent cannot perform any of its tasks,

Or, the communication with the agent is up but:

- The agent can only perform some of its tasks
- The agent cannot perform any of its tasks

All of the mentioned failures of agents are critical, because MAS are vitally depending on the agents' capability to communicate. The system disturbances include exceptions like rush orders, quantity and mix variations, or incorrect deliveries, which should be considered in the system concept. A failure tolerant system should be able to absorb all these failures mentioned above. However, since the system disturbances could be generally handled with a well designed architecture and defined related agent behaviours, we will focus in this study on physical component and software entity/agent failures, which are mostly unpredictable and evolve dynamically. Besides, our attention will be pointed out to failure tolerance in case of the complete breakdown of particular physical components or failures when agents cannot perform any of their tasks.

5.3 The Failure Handling Concept

Figure 5.1 shows the different types of agents. One the left side we have agents that supervise particular resources (e.g. conveyors or robots). These agents are linked with the LLC based on IEC 61499, which is used to define and program a LLC application that controls the physical system, acquires the information from sensors and actuators, and enables the agent level to perceive the environment through it. On the right side we have strictly functional agents without reference to a special resource of the system (e.g. contact-, supply-, or order-agent – as explained in the previous section). As said before, we distinguish physical component failures (Failure Type LLF, Low-Level-Failure) and software failures (Failure Type HLF, High-Level-Failure). The failure type LLF represents failures of the hardware or within the associated low-level control. Such failures could be:

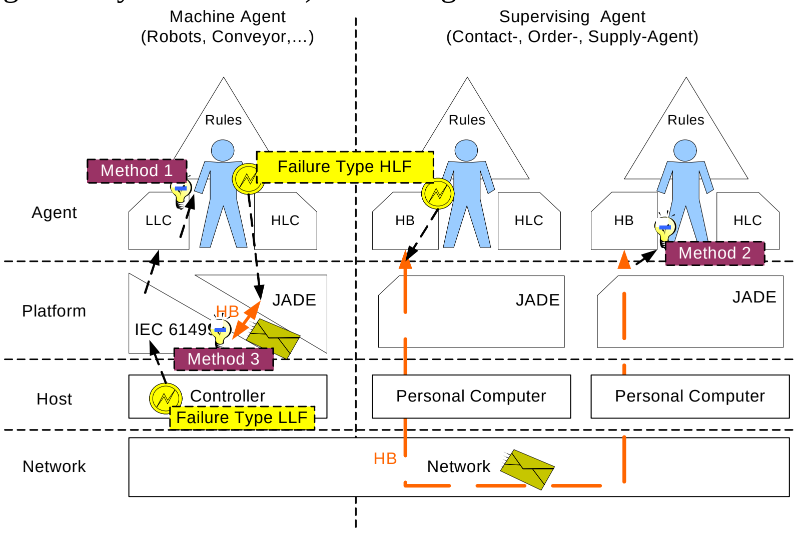


Figure 5.1: Architecture and related failure types

(a) the breakdown of conveyor belts, which means within the actual hardware, or

(b) a failure in the controlling function block based application.

In case (a) the LLC is able to recognize the failure of the hardware via sensors and can inform the agent through specially implemented *statusChannels* between LLC and HLC (defined for the exchange of status messages). As a result, the agent reacts in an appropriate way, based on its predefined failure rules [Mer09]. In case (b) the hardware is potentially uncontrolled and the agent cannot get any error message from the LLC. However, due to the implemented heartbeat between the LLC and HLC, which could be compared with a watchdog (that is broken in this particular case), the HLC of the agent recognizes the failure of the hardware controlling functions and is also able to react on it. The existence of the *statusChannel* is of vital importance since it enables the mutual and periodical supervision between LLC and HLC. If one part is not responding within a predefined time, the other part starts failure handling procedures. Hence, the agents' reaction procedure is nearly the same as in (a), because the agents only recognize physical component failures and start failure rules.

The failure type HLF targets unexpected failures of software agents. In this case an agent itself is not able to perform any of its tasks. Generally, this failure detection is done with a heartbeat (HB) mechanism, where two agents periodically exchange messages. In large and complex multi agent systems this could lead to a huge amount of exchanged messages, which could affect the system in a negative way and further lead to a weak performance. A way to keep this message exchange in a low amount is to use this form of heartbeat mechanism only for associated agents. Therefore we distinguish between two types of agents based on their types of failures:

(c) a failure of HW-controlling agents, and
(d) a failure of functional agents.

The HW-controlling agents use an internal heartbeat to monitor a failure in case (c), implemented between LLC and HLC. Hence, the LLC-part of the agent, which is running in IEC61499, is now able to identify the failure within the JADE-part of the agent via this heartbeat. It starts a procedure, which informs the Agent Management Service (AMS) of the platform, to restart the HW-controlling agent completely. This is ensured with a backward-recovery, where the agent's actual knowledge base is used to get all necessary information, like name, address, or goals of the agent. Therefore, firstly the erroneous agent will be killed and removed from the platform and then a new agent with the same representation will be started. In this case the agent's knowledge base, which covers its actual internal representation, is uploaded from the system memory or agent backup files.

On the other side, functional agents don't have the possibility of such an internal watchdog, because there is only one SW-Module (JADE). Therefore these agents use the common heartbeat mechanism. This is done by periodically sending messages through the JADE runtime. In case of the presented system, this message exchange is done between the CA and the OA and SA for each received customer order.

5.4 Implementation

a) Heartbeat-Mechanism

At the beginning we will focus on the heartbeat mechanism in general. Normally a heartbeat mechanism is based on the exchange of request and answer messages between two entities. Figure 5.2 presents the heartbeat between two agents. T_{HB} is the period of each heartbeat. The period should be longer than the expected heartbeat time t_{HB}, which consists of the reaction time t_R and the times necessary to transmit both messages, the request and their answer, over the network. The reaction time is as long as an agent needs to receive and process the message including the time to prepare an answer message.

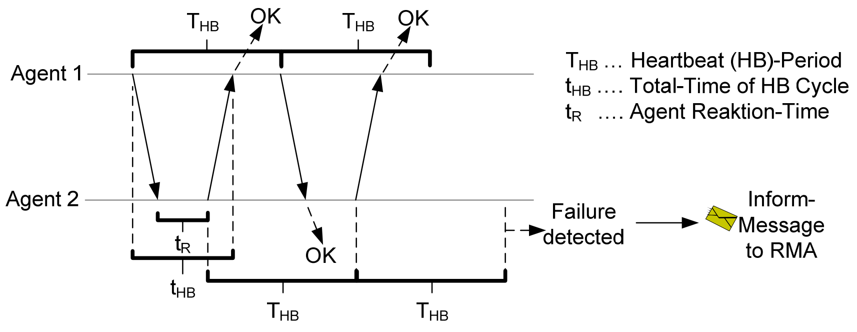


Figure 5.2: Heartbeat mechanisms

When an agent starts the heartbeat mechanism, it is awaiting the answer in a given time (in our case T_{HB}). When a correct answer message is received in the specified time, the agent knows that the other one is alive. The picture shows, that if an agent receives a request, it replies, awaiting again the request message within T_{HB} time. That way, it is possible for two agents to control each other.

b) Heartbeat-Organization

If one agent sends no answer in the given time, the other agent realizes this missing message as a failure and informs the platform managing agent about the failed agent. This offers many possibilities to implement a heartbeat mechanism in multi agent systems, as shown in Figure 5.3. All the proposed solutions, as presented in Figure 5.3, focus on the already introduced machine-agents, while the functional agents always use a peer-to-peer mechanism for realizing the heartbeat. One possible solution – shown as broker architecture – is a centralized version with one supervising agent (CA), which is the heartbeat-partner for every other MA in the system. The major drawback is that due to the centralized aspect this agent could become a bottleneck, which leads to unreal failures caused by timing problems. A second aspect is the significant amount of messages needed to realize such a heartbeat mechanism [Kop09]. The second solution is to create peer-agents, which are only responsible for each other. In that case, one agent (e.g. CA) is necessary to manage the peers at startup and introduce agents before a heartbeat messages are exchanged. The bottleneck is reduced to the startup-phase of the agents. Our proposed solution is a hybrid architecture, where only the functional agents perform a peer-to-peer like heartbeat mechanism by sending messages over the network. The machine agents perform the

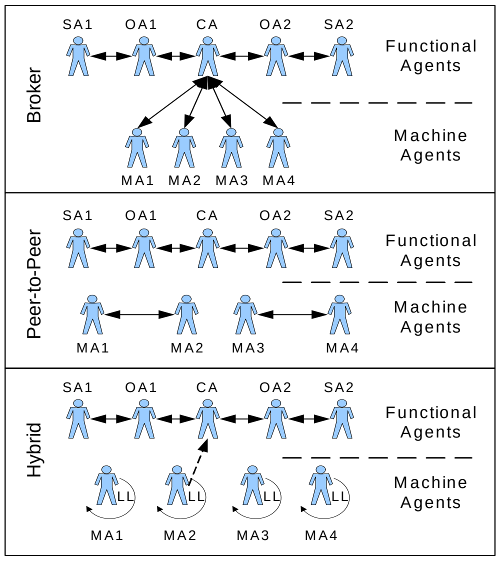


Figure 5.3: Heartbeat scenarios

heartbeat mechanism without sending messages over the network. They use an internal heartbeat between their HLC and LLC. Nevertheless, in the case of the complete breakdown of both (HLC and LLC), this failure could only be detected from its surrounding agents while they interact with it. This could be solved through a time-constrained message interaction or through special sensors and will be a part of our future research being mentioned here only to meet the principle of completeness. We focus more on the possibilities of a hybrid heartbeat using a heartbeat communication between LLC and HLC.

c) Heartbeat Communication between LLC and HLC

The sanity of both parts of MAs is detected by heartbeat messages exchanged between the two layers. In our implementation the message content is separated from the message transfer channel. Since both layers are implemented in different runtime environments, the initial realization is done via local network communication, as already described in [Kop08a] [Heg08]. The bidirectional `bStatusChannel` has the task to provide status information of the mechanical component and the real-time capable LLC-layer to the HLC. Since the payload on this channel is minimal and recent information on the component-status is essential for the HLC, this channel is used to exchange the heartbeat messages. The heartbeat functionality of the LLC is realized by a small function block (FB) network containing 3 FBs as shown in Figure 5.4a. The composite function block (CFB) “bChannelStatus” is actually responsible for receiving and sending the heartbeat from and to the HLC. It contains the necessary service interface function blocks (SIFBs) for the communication within the controller. SIFBs are designed to provide the required interfaces concerning communication with remote resources as well as access to hardware elements (such as physical I/O ports of a controller) of a device. The CFB “LLC_HeartBeat” contains a network of 3 FBs (see Figure 5.4b) that are responsible for triggering and observing the heartbeat. After initialization of the LLC, the heartbeat generation is started by the START-event received by the “E_CYCLE”-FB. This block now triggers “bChannelStatus” to send a heartbeat to the HLC every 5 seconds. Simultaneously a flip-flop is set as a confirmation for sending the heartbeat (FB “E_SR”). This flip-flop is reset as soon as the HLC returns the heartbeat. A response of the HLC heartbeat within 5 seconds indicates that the HLC is still functioning. However, if the heartbeat is not returned by the HLC within 5 seconds, an error-event is triggered and sent to

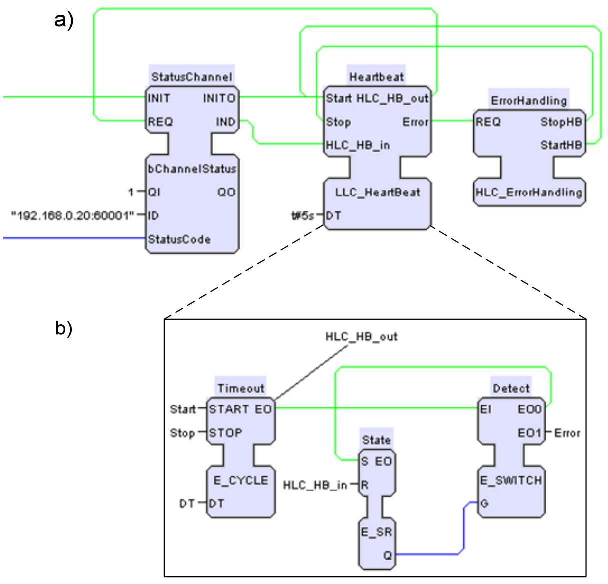


Figure 5.4: Low-Level heartbeat functionality

the CFB “HLC_ErrorHandling”. In case of a detected error, “HLC_ErrorHandling” stops the heartbeat and is able to activate recovery measurements. For instance it can send a message via the network to the remote management agent (RMA) with the content that the HLC is no longer reachable. Or it may contain SIFBs with functionalities concerning the operating system (OS) of the controller. In this case it may trigger a restart of the HLC agent platform or even a complete restart of the controller. As soon as the failure recovery measurements are executed, the heartbeat is restarted [Kop09].

The failure recovery procedure, when the HLC doesn’t receive a heartbeat from the LLC or receive the message from it that the related hardware component is out of order, will be presented in the remaining part of the chapter. We tested the behaviors and reconfiguration abilities of the presented knowledge-intensive multi-agent architecture using simple transport tasks and introducing some internal disturbances (failure of particular components e.g. conveyors).

5.5 Reconfiguration Abilities of the KASA

The route planning and best path algorithms are highly researched topics in computer science for many years [Tom05]. Based on Dijkstra’s algorithm, we implemented a simple, but very efficient Shortest Path Algorithm (SPA), which is used by diverter agents for calculating a routing table and by pallet agents for the distance calculation during the negotiation process regarding the transport allocation [Mer08]. Furthermore, we introduced a Change-Direction-Algorithm (CDA) to test the *failure recovery* characteristic of our architecture. It handles a breakdown of conveyors which might lead to unreachable destinations (machines). The CDA is able to find the best stable configuration of the transport system by changing the directions of specific conveyors [Kop08b]. The CDA, like the SPA, starts from the ontology representation of the proper functioning system and presents its structure as a matrix. The matrix is then filled with all intersections and adjacent conveyors. After that, the occurred failure is introduced in the matrix and the CDA starts to change virtually the direction of all potential-solution conveyors and checks the system functionality. The new solutions will be compared with previous ones and saved, if marked as better. However, the new solutions could also effect and cause the direction changes of some adjacent conveyors in the system. The CDA is therefore programmed as a recursive algorithm in order to being able to evaluate such states as well. The system functionality in a dynamic environment is demonstrated by introduction of failures of particular resources. Correlated to our concept, the system environment is represented within the ontology. The instances are used to represent locations whose attributes and relations provide details of locations reachable from it, as suggested in [Cor05]. The path is being defined as a set of segments (conveyors) between the current position and the destination. The constraints of system components (e.g., conveyor overloaded – busy) are also incorporated in the ontology and regulated trough appropriate agents’ behaviors. A representative part of the pallet transfer

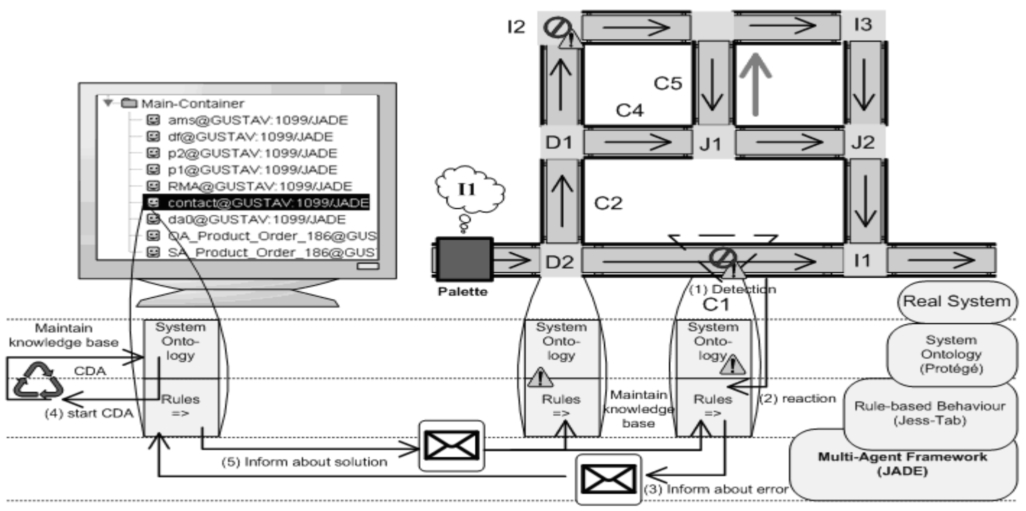


Figure 5.5: Simplified Scheme of the Transport System and MAS Framework

system is shown in Figure 5.5. The presented section consists of Diverters (D1, D2), Junction (J1, J2), Index Stations (I1, I2, I3) and Conveyors (C1, C2, etc.) between them. The main objective of the system is to transport pallets using the shortest way to their target destinations. In the presented example the destination is I1. In the case of a proper system function, the shortest path is via the intersection D2 and the conveyor C1. However, in case of a system failure, for example the break down of the conveyor C1, the pallet will not be able to reach its destination in the usual way. To handle this, the system has to be able to react on the new state. The ability of the system to change from its current configuration to another configuration—possibly without being taken off-line—is of highest importance. In case of presented system failure the system reacts as follows:

1. The hardware of the system detects the failure using sensors. The low-level control informs the corresponding MA through the low-level communication interface.
2. Based on the given information the agent updates its knowledge base and sends the “INFORM” ACL message to all related agents (diverter and neighbor agents) about the detected failure. Each diverter has to recalculate and update its routing table.
3. Furthermore, the agent also sends a “FAILURE” message to the CA, which is equipped with the CDA algorithm, in order to check the system functionality.
4. The CA starts the CDA and compares its results with the actual system state.
5. In the case that the CDA recommends a new configuration, the CA updates its ontology, sends “REQUEST” message to related conveyor agents to change directions, and “INFORM” messages to diverters, updating the system representation.
6. Each diverter will recalculate and update its routing table. Having accurate information and an up-to-date world model of the system is of primary importance for diverters. Due to their role to receive pallets coming from input conveyors and—according to their destinations—to route them to the appropriate output conveyors.

The reaction of the system on the failure of the index station I2 will follow the same procedure. However the CDA notices that the index station I3 is not reachable any more and suggest the direction change of the conveyor C5.

5.5.1 Simulation Approach

In order to test the effectiveness of the presented approach we have used the MAST, which is able to provide agent-based simulation support for our empirical study and the TMS is used for automatically running predefined sets of test cases described in XML files. A total set of 52 test cases were generated from the scheduling strategies as input to the TMS. Each test case has a workload of 7200 customer orders and applies FCFS (First Come, First Served) workflow scheduling strategy. A customer order consists of a particular product type to be built and a randomly generated due date that has to be respected. We defined three product types (simple, medium or complex) which differ in the type and number of machine operations needed to assemble the final product. The type of machine operation defines the related machine that has to be visited to accomplish the operation. To finish a particular product all related machine operations have to be completed and the product has to be delivered to the storage. The shift time for a test case was set to 24 hours in order to ensure that randomly selected customer orders could not easily be finished in the study context without an effective use of all available resources (same as in the chapter 4). In order to test the influence of the pallet jam, the performances of the MAS are also tested when different numbers of pallets have been available (5, 10, 15 or 20) for each test case respectively. Moreover, being aware that the failure duration as well as the position of the failed conveyor for the overall system could play a significant role on system output, we considered also these during the test case definition. We classified the risk of a failing conveyor in 4 failure classes:

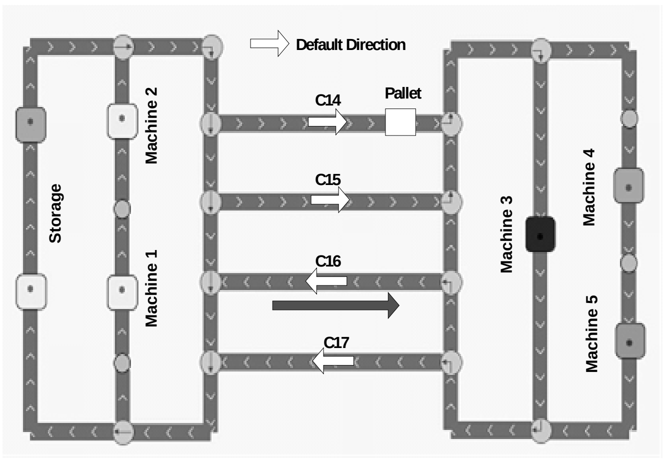


Figure 5.6: Test bed Layout

a) test case without conveyor failure;
b) test cases, when one conveyor failed (C14) but all machines are reachable over redundant conveyor (C15) as presented in Figure 5.6;
c) test cases, when two conveyors failed (C14 and C15) but the CDA is started and while conveyor C16 changes its directions the right side of the system becomes reachable again,
d) test cases, when two conveyors failed and one part of system is not available for a particular amount of time.

Test cases with failures are further defined considering different duration of particular failures (15 minutes, 1 hour, 2.5 or 5 hours) respectively. Each failure type is introduced after 5 hours of uninterrupted work.

5.5.2 Simulation Results and Discussion

We use the number of finished products as reliable measure for efficiency of the system. The dependence of the system output on the number of pallets, failure type as well as failure duration is presented in Figure 5.7 (refer to Table 5.1 for more details). The diagram shows that the number of finished products proportionaly increases with the number of used pallets. Even when the conveyor C14 fails for a short period of time, due to the existence of the redundant conveyor C15 that takes over its duties, this does not significantly influence the system performance. Nevertheless, this is the case when the number of used pallets is relatively low. Increasing the number of pallets over 15, the conveyor C15 becomes a bottleneck causing pallet jam and reducing the transport flow. This is especially the case when 20 pallets are used. Although, more pallets have been introduced, due to overload the system showed worse performance than when 15 pallets are used. In the case, when two conveyors (C14 and C15) fail at once and the right side of the transport system becomes unreachable, our system starts to reconfigure itself and automatically switches the direction of the conveyor C16 in order to function properly.

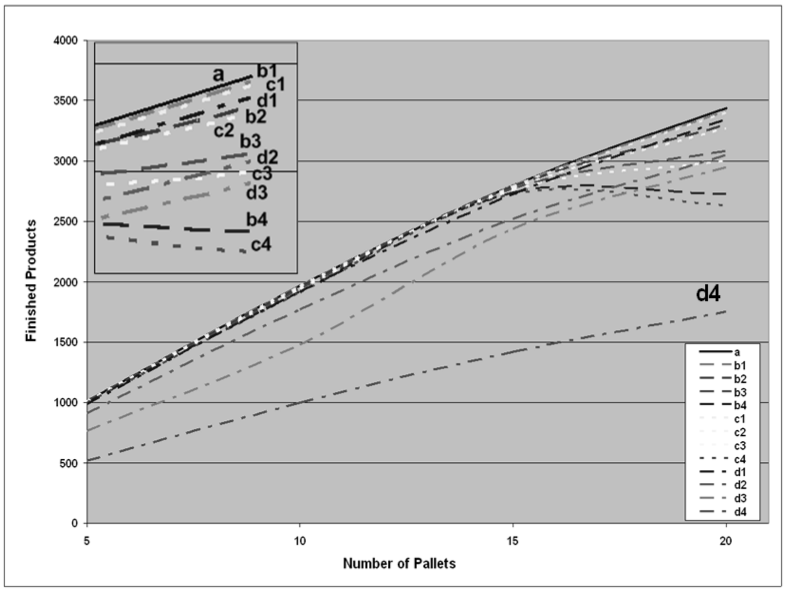


Figure 5.7: Number of finished products within a test case

Failure Classes		Number of Pallets	5	10	15	20
	Test case	Failure duration				
a) No Failure	a		1011	1965	2797	3440
b) Redundant Conveyor Failure	b1	15 minutes	1010	1963	2792	3410
	b2	1 hour	1008	1957	2785	3296
	b3	2.5 hours	1003	1941	2768	3078
	b4	5 hours	998	1915	2737	2724
c) CDA	c1	15 minutes	1009	1961	2786	3398
	c2	1 hour	1005	1954	2778	3269
	c3	2.5 hours	1002	1936	2761	2998
	c4	5 hours	996	1910	2723	2629
d) Part of system unreachable	d1	15 minutes	986	1916	2726	3342
	d2	1 hour	910	1768	2517	3044
	d3	2.5 hours	760	1476	2433	2945
	d4	5 hours	513	996	1419	1750

Table 5.1: Number of finished products considering the Number of Pallets and Type of Failure

Nevertheless, this causes even more pallet jam than in the previous case (b), while C17 cannot be used as redundant conveyor now, since C17 and C16 have to overtake the whole transport load from one side to the other of the transportation system. The system performance increases also here with the number of used pallets until the optimal number of pallets (for the presented Test bed layout this number is 15) is being reached. However, due to the traffic jam the system produced 2.6% products less operating within this state for 5 consecutive hours. Consequently, when 20 pallets are used the number of finished products is decreased up to 23%, in comparison to the state when everything is functioning well. Nevertheless, the application of the presented CDA algorithm significantly improves the system efficiency of MAS when compared to the performance of conventional systems, which are presented in case d), and this particularly in the cases when the duration of failure is significantly long. Being able to automatically reconfigure the system layout, the presented MAS architecture outperformed conventional systems by producing 11.8% products more when optimal number of pallets is used and when the duration of the failure is 2.5 hours or even by 46.6% products more when the conventional system has to wait 5 hours on the external intervention. We are aware of the facts that diverse Test bed configurations, the time of the failure occurrence, combination of different failure types as well as the order of their appearance could result in different system performance from the presented and this is the reason we will consider these factors in our future work.

5.6 Summary

A specific failure scenario is applied to show the reconfigurative behavior of the system. By reconfiguring the behaviors of its physical components the system is able to ensure its global functionality even in the case of several component failures. The simulation results indicate that our approach is well suited for building complex systems to enhance the system efficiency. Moreover, even though the failures are induced in this implementation only in the KASA simulation environment, the system's ability of selfreconfigurability is existent also for the real physical system. Only the graphical simulation would have to be substituted by the physical equipment resulting in minor changes to the LLC but without having to change the structure and reasoning of the HLC.

6. Simulation of the KASA Environment

"He that would perfect his work
must first sharpen his tools."
Kung Fu-tzu Confucius

6.1 Introduction

Simulation is a powerful method for performance evaluation and quality improvement of designed control solutions. Combining the agent based modeling and simulation, it is possible to tune the system behavior and choose an appropriate design before actually implementing it in the real system [Vrb05]. The areas in which such approaches are already in use, for instance testing the manufacturing scheduling, simulation of the packing cell, simulation operations in a rough mill, simulation of the manufacturing supply chain operations, etc., are previously presented in the second chapter. However, the manufacturing systems emerge and evolve very fast and exchange of accurate information and knowledge between entities within these systems is of highest importance, since the decisions made have to be based on exact information. The main limitation of the approaches mentioned above is the lacks of solutions for covering such dynamic information exchange.

The importance of ontologies for development of intelligent knowledge based systems could be seen trough support of interaction mechanisms, insurance of interoperability between agents, re-use of knowledge and simplification of the solution development [Nec91, Hob87]. The technical challenges and benefits of using ontology-driven approach for simulation modeling, e.g., such as role in conceptual model design phase or role in simulation integration and simulation composability, have been described and presented by Miller and Baramidze as well as Benjamin et al.. However, these advantages are still not really explored by modeling and simulation communities [Mil05, Ben06].

6.2 Tools

We used MAS as a simulation model to build scalable and flexible manufacturing system. MAS is recognized as a flexible and reusable modeling framework which enables rapid development of customized decision solutions for manufacturing [She00]. As said before, our simulation model employs the ontology concept represented in the OWL, which serves as a standardized mean for describing ontologies. We use Protégé-2000 [Stanf07] as an integrated software tool to develop both the ontology and the knowledge base. Protégé-2000 is an open-source ontology and knowledge base editor designed by Stanford Medical Informatics at the Stanford University of Medicine. It is implemented in the Java programming language and can manage ontologies in XML, XML Schema, RDF(S), DAML+OIL and OWL. Protégé-2000 provides a plug-and-play environment and can easily be extended. Ontologies in Protégé-2000 can be handled through a friendly graphical user interface. We use Protégé ontology editor to design the ontology, create related instances of classes, and define attribute values (Figure 6.1).

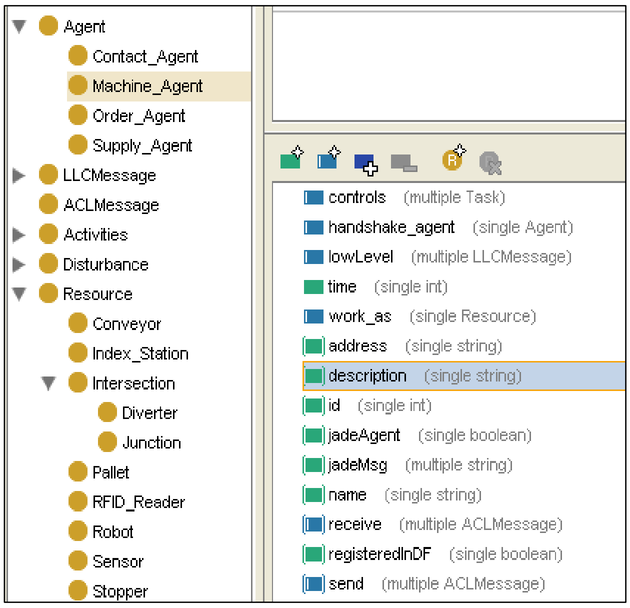


Figure 6.1: Part of ontology (Machine Agent)

The reasoning is implemented using the Jess expert system shell [Sandi07]. Java Expert System Shell (JESS) is a small and very fast rule engine developed by Ernest Friedman-Hill. It is based on CLIPS [CLI09] but written entirely in Java. JESS is a tool used for building the rule-based expert systems, which can be seen as a set of rules that can be repeatedly applied to a collection of facts about the world. In this context, a fact is identified as a construct that defines a piece of information that is known to be true. Rules are simple statements that consist of an if-part and a then-part. Jess applies a special Rete algorithm to match the rules to the facts. When the particular input information, which is coming from the environment, matches the facts in the if-part of the rule, particular actions, which are defined in the then-part, are executed. However, this can deduce new facts that could be a reason for firing of some new rules. Compared to the imperative one, the declarative knowledge is more reusable and modular, has better semantics and makes detecting and correcting contradictory knowledge easier [Gun04].

We used a plug-in called JessTab to integrate Protégé with JESS [Eri03]. JessTab integrates these two tools by mapping Protégé instances to Jess facts and enables the JESS rules to fire when the appropriate conditions in the Protégé knowledge base are met. Using this plug-in, JESS can directly manipulate the ontology and instances and infer new facts deduced from them.

The presented multi-agent architecture is built using the Java Agent DEvelopment Framework (JADE). The JADE platform is developed by the Telecom Italia Lab, in compliance with the standards defined by FIPA (Foundation for Intelligent Physical Agents), which is a non-profit organization involved at producing standards for the interoperation of heterogeneous agents. JADE is written entirely in the Java language, which simplifies the implementation of the presented multi-agent system with other tools mentioned above. The JADE platform enables each agent to manage its own life cycle, register its services, search for agents providing particular services, discover them and communicate with related agents. The JADE architecture enables agent communication through message exchange using the agent communication language (ACL) [Fou03] that is based on the speech act theory [Aus62].

This agent platform can be also distributed across multiple hosts, where the inter-agent communication is managed by exchanging ACL Messages. Several JADE platforms can be indirectly connected over the DF Federation [Kop08f]. More information on JADE can be found at [JAD08]. The agents reasoning capabilities are incorporated into the agents using JESS. The integration of Jess component into a JADE agent is done by instructions written by Cardoso [Car09]. For the debugging and analysis of communication between agents and correspondingly for the manufacturing workflow observation, we used a tool called the Java Sniffer [Tic06]. The Java Sniffer is a stand-alone Java-based agent-communication visualization tool, developed by Rockwell Automation, Inc., which. can be easily attached to running JADE system. It receives messages from all involved agents in the system and presents it from different points of view (Figure 6.2). The Java Sniffer supports the resolving of communication problems in system during development phase.

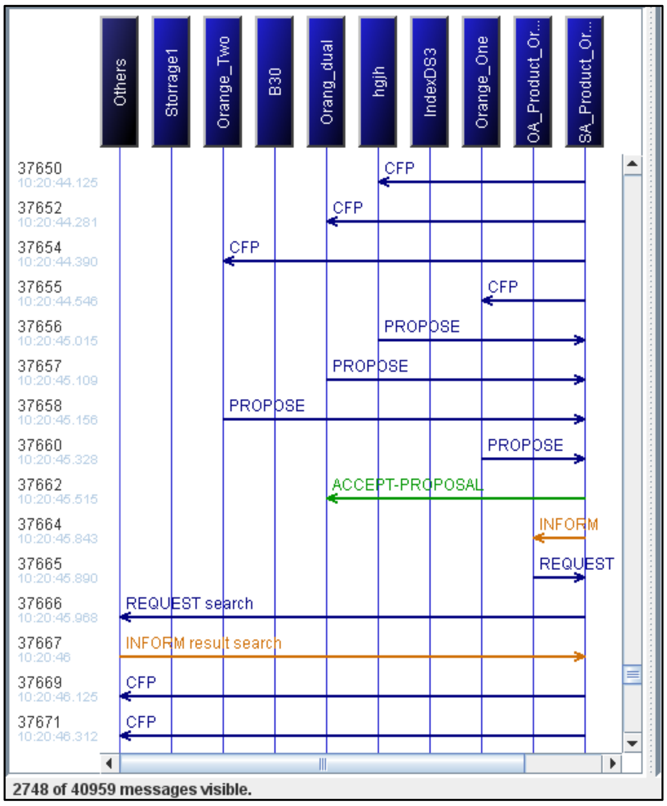

Others
Storrage1
Orange_Two
B30
Orang_dual
hgjh
IndexDS3
Orange_One
OA_Product_Or...
SA_Product_Or...
37650 10:20:44.125 CFP
37652 10:20:44.281 CFP
37654 10:20:44.390 CFP
37655 10:20:44.546 CFP
37656 10:20:45.015 PROPOSE
37657 10:20:45.109 PROPOSE
37658 10:20:45.156 PROPOSE
37660 10:20:45.328 PROPOSE
37662 10:20:45.515 ACCEPT-PROPOSAL
37664 10:20:45.843 INFORM
37665 10:20:45.890 REQUEST
37666 10:20:45.968 REQUEST search
37667 10:20:46 INFORM result search
37669 10:20:46.125 CFP
37671 10:20:46.312 CFP
2748 of 40959 messages visible.

Figure 6.2: Java Sniffer: Negotiation between Supply and Machine (Robot) agents

6.3 System Integration

In order to present the function of each software component of the system, we will observe activities of a representative part of the Transport System (Figure 6.3) in the case of a failure. It consists of diverters (D1 to D3), a junction (J1 and J2) and an index station (I1) interconnected by conveyor belts. The shortest path for the palette to I1, when all system components are functioning correctly, leads over D3 and J1. Considering that each diverter has accurate world model based on which it can calculate the shortest path, the system is going to route the pallet using this way. However, in case of a system failure, for example the break down of the conveyor belt between diverter D3 and junction J1, the pallet will not be able to reach the destination using this optimal way. To handle this exception, the system has to be able to reconfigure itself — to change from its current configuration to another configuration — without being taken off-line. For this particular case, the diverter D3, has to be:

1 informed about the failure, and
2 able to maintain the knowledge about the system and recalculate the shortest path data table based on this new information.

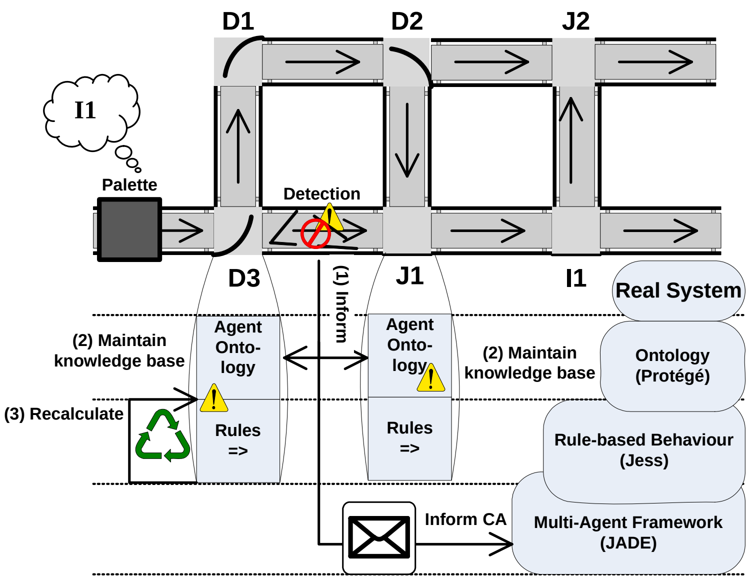


Figure 6.3: Simplified scheme of the Transport System

The information about the failure is sent to all related system components by CA, which is informed from corresponding MA that supervises the broken component. To enable faster system reaction, this MA informs usually also neighbor agents, as described before in section 5.5. All agents that receive this message should handle it in a similar way by maintaining their own knowledge base according to the received message and re(-acting) appropriate, when defined with rules. As a result the diverter D3 is able to reroute the pallet over the newly calculated route D3-D1-D2-J1-I1, thus preventing that the palette is stuck at junction J1 until the broken conveyor belt is replaced or repaired.

To face this, every agent has to be able to sense and understand its environment (represented in the system-ontology), as well as to (re-)act on these inputs with a specific behavior (e.g. inform others by sending messages). This representation, where every part of the system is stored as an individual, is the fact base for the JESS engine. The JESS-engine maps all the instances of the system-ontology and handles them as facts. Based on these facts, the left hand sides of the agent's rules, mentioned in the section before, will be evaluated thus activating the actions on the right hand side or not. This rule based agent behavior for handling the system evolution and for reaching the agents' aims, which is coded in the JESS-language, is one part of the agent behavior. The other part is the communication behavior which is influenced by the multi-agent framework JADE. JADE provides the agent lifecycle and the message transport service for sending ACL Messages. As any other class, each agent is stored as an individual in the ontology and as a fact in the JESS-fact base. Together with the JADE representation of the agent (JADE-Agent), this is the second representation of each agent (JESS-Agent) and we have to take care, that changes in one representation impacts the other in the same way [Car09]. To face this, we bind the JADE Agent to the JESS-Agent and integrate the JESS behavior into the JADE behavior like presented in Figure 6.4. Through this, the JESS-engine of the agent including the rule-based behavior is linked to the JADE agent cyclic behavior. This behavior enables each agent to fire its rules according to their fact-

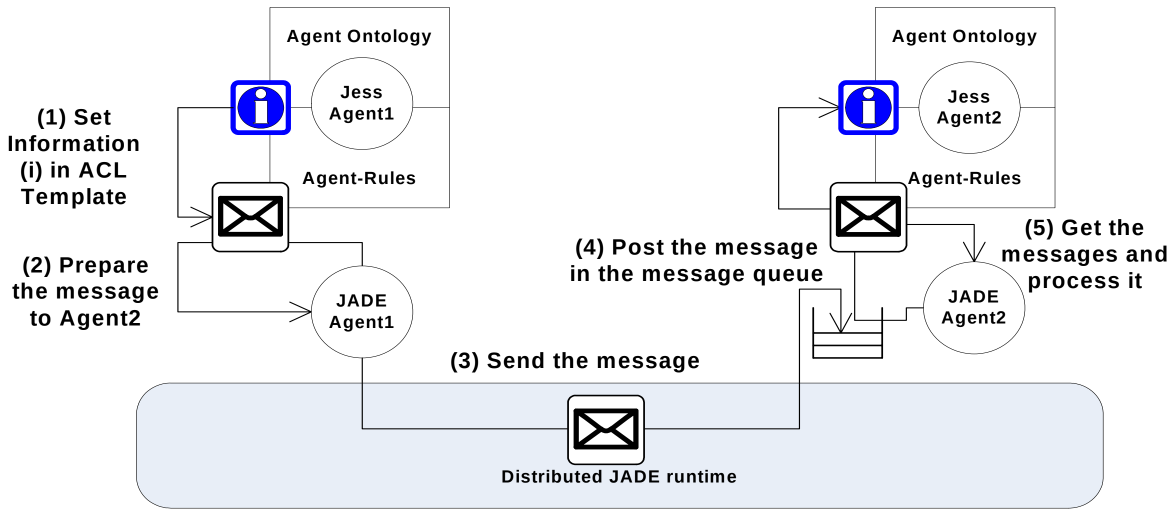

Agent Ontology
Jess Agent1
Agent-Rules
(1) Set Information (i) in ACL Template
(2) Prepare the message to Agent2
JADE Agent1
Agent Ontology
Jess Agent2
Agent-Rules
(4) Post the message in the message queue
(5) Get the messages and process it
JADE Agent2
(3) Send the message
Distributed JADE runtime

Figure 6.4: Message conversion between JADE framework and JESS-Agent

base. As the communication between the agents continues, the fact bases of the agents are changing permanently. Each agent has rules to handle incoming messages and modify their ontologies.

In the case that a JESS-Agent has to send a message to other agents, the system works as follows:

1 The content of the message will be derived from the ontology and a Jess-rule will store it in an ACL-message template. This template is the interface between the ontology-representation of the message and the message processed by JADE.

2 A special send-behavior enables the JADE-Agent to prepare the message for sending through the distributed JADE runtime.

3 In this step the ACL-message will be sent from one Agent to another Agent, independent from their location—either within one JADE-container, between two containers in the same platform or between different JADE-platforms.

4 The message arrives in the message queue of the receiving JADE-Agent.

5 It is processed and stored in the receive-slot of the Jess-Agent. For the purpose of allowing asynchronous receiving and processing of more than one message, the receive slot is implemented in form of a multislot. Otherwise, as experienced, message could get lost (deleted with other incoming messages).

6 After receiving a JADE-ACL-Message the JESS-Agent maps this message automatically to the agent's fact base, what causes that agent switches from the blocked state into the jess-behaviour-state and is able to fire its rules according to newly created facts. This behaviour is supported by the JADE function for receiving messages and it is essential for the system, otherwise the agent wouldn't have the facts to fire on it.

The messages are being exchanged between agents all the time, since the system is constantly changing its state and all system components have to have up-to-date information about its current status.

6.4 MAST Simulation

For the empirical studies of the system architecture performances presented in the 4th and 5th chapters, we used MAST simulation and Test Management System, whose features are in-depth described by Vrba et al and Merdan et al. respectively [Vrb08, Mer08d]. Considering the used JADE framework and similar architecture, we are further streaming the integration of the KASA and MAST simulations. The fundamental steps, which are already done in this direction, are presented in [Mer08e].

6.5 Summary

Simulation is a very effective way for testing different control architectures and improving quality of designed solutions. In our case the simulation has been used as an indispensable tool for tuning and validating the agents' behaviour and knowledge. The most important aspect of this agent-based simulation is the possibility that the agent-control algorithms developed for the simulation model could be reused, in ideal case without any modification, for the actual real-life control. Using MAST enabled us to imitate the complexity of the real system in the simulation and to test different scheduling strategies much faster then in the real testbed. Moreover, the relations between different manufacturing parameters, such as throughput, tardiness, capacity, complexity of products assembled, could be established much more effective, at lower costs, and – what should not be underestimated – safer. We are able to confirm that the agent paradigm is suitable for building, modeling and simulating complex manufacturing systems.

7. Conclusion

The current manufacturing control systems respond weakly to the emerging challenges caused by new technological developments and market demands. In order to stay competitive in the dynamic global market these systems have to be able to effectively react to sudden changes in customer demands, constant evolution of technology, and unpredictable events such as failures and disruptions. However, currently applied centralized control approaches are good in optimizing production but weak at responding to change, mainly because their rigid hierarchical structure.

Intelligent agents offer a new distributed control approach that, using concepts of autonomy and cooperation, leads to more flexible and robust production systems. Agents with specialized expertise and high level of autonomy cooperate together to accomplish individual, as well as system objectives. Although confirmed as a promising approach and deployed in a number of different applications throughout the last few years (e.g. MAST, Production 2000+, etc.) the widespread adoption of agent-based concepts by industry and governments is still missing. As the main reasons identified, upon others, are the lack of awareness about the agent technology potentials, missing trust in it as well as lack of standards, methodologies and development tools that would simplify the integration of this technology in the manufacturing domain. We further presented challenges that should be solved in order to accelerate the adaptation of agent-based technologies. Achieving interoperability in the heterogeneous manufacturing environment, the transformation of received raw data into knowledge, the linkage of the agent-based system to real-time information and its integration with the FB-technology as well as support of principles such as generality, reusability and long-term usage are defined as main challenges whose elucidation could further lead to the higher implementation rate and raise confidence in this technology respectively.

To face requirements mentioned above we developed knowledge intensive multi-agent architecture applied in demanding assembly domain, where each agent has its own "world model" of the environment. The ontology is used to formalise agent knowledge and to describe the concepts and relationships that can exist within a multi-agent system. Incorporating semantics into the data, an ontology specifies the meaning of terms which are used during communication enabling knowledge interoperations between agents. Ontologies are also used to record actions and events as an explicit knowledge so that they can be analyzed afterwards. In the frame of this thesis we developed persistent ontology able to support knowledge exchange during the entire manufacturing process, from the ordering over production until the final shipment to a customer.

As a basis for simulated architecture, we use the "Test bed for Distributed Holonic Control" at the Institute for Automation and Control, Vienna University of Technology. Considering the Testbed and defined layered manufacturing structure we created related agent classes. The established multi-agent architecture, focused on clear decentralization of the manufacturing system, was able to handle complexity of the used Testbed layouts during the simulation tests reaching good system performances and increasing its agility compared to the present state of the art systems. This was especially notable in the planning process, where upon others, the ontological representation of the product model enabled agents to autonomously reason about the used concepts, linking automatically between product models, production processes and production equipments. Moreover, the ontology-based system model facilitated coordination between agents making system knowledge both machine-interpretable and shareable at the same time. Besides, the implemented interface between agent and low-level as well as its ontological representation provides agent with ability to “understand” the exchanged messages with low-level control and correspondingly the acquired information from sensors and actuators.

Considering the scheduling of production resources as one of the key features in the current competitive and dynamic manufacturing environment, we have done extensive tests applying our architecture and proving its effectives and efficiency. In order to create flexible scheduling able to cope with conflicts derived from the resources sharing among the production orders, we combined multi-agent approach with negotiation mechanisms for task allocation, where additionally each resource agent performs local scheduling using dispatching rules. The simulation study proved that the system was able to handle the scheduling and production process on its own. The measured system performance was particularly improved when we augmented the scheduling calculations to explicitly consider the transportation durations between the machines. We also measured system robustness by systematically comparing the overall system performance (e.g., number of finished products) when using one of four re-scheduling strategies in case of machine disturbances/failures. In the empirical evaluation the Complete Rerouting re-scheduling strategy outperforms all other strategies significantly.

Since the concerns regarding the stability and survivability of multi-agent system, especially in unpredictable environments, are mentioned as one of the key reasons for weak implementation rate, in the fifth section we tested system failure tolerance, i.e., its ability to detect, isolate and recover from failure. We presented our failure tolerance approach that combines the heartbeat mechanism for failure detection and the supervisor agent approach for system failure absorption and recovery. The simulation results confirmed also system reconfiguration abilities and indicate that our approach is well suited for building complex systems to enhance the system efficiency.

Bearing in mind the advantages of presented multi-agent architecture, we are aware the fact that this technology has to mature through real industrial applications, to establish multi-agent system’s ability to autonomously and faultlessly govern the entire manufacturing systems. On the one side we think that the agents’ ability to maintain an accurate internal representation of pertinent information about the environment in which it operates has to be further developed. However, this could significantly improve its self-monitoring and self-control capabilities. In this regard it is of vital importance to define the constraints that the subsystems (e.g., sensors, actuators and operator control units) place on symbolic world model representations as well as to specify the means to measure the quality of ontological representation for autonomous agents.

On the other side, the capability of enterprises to form network organizations and cooperate with partners is an important factor for competitive market position. The information and knowledge exchange between partners play a critical role for the success of such networks. It is of highest importance to have an optimized information flow to find the appropriate knowledge source in the desired quality and in adequate time. In current networked organizations it is usually not transparent to the partners, which knowledge is available at which partner's site. Our proposal is to use semantic technology together with software agents in order to improve knowledge capturing, knowledge reuse and knowledge transfer in such networks. The development of the multi-agent architecture able to govern and support such networks is one of the biggest challenges. Moreover, considering the extremely heterogeneous nature of such environments the more serious work has to be done in the field of an ontology engineering e.g. merging and mapping.

Finally, as successors of the idea that the best multi-agent system is the implemented one, we will continue further to stream its real life application. We are currently developing the emulation for each particular component of the transport system and setting the basic preconditions (such as porting the selected software platform—JADE and JESS—to the embedded target platform) for the deployment of the developed MAS into the real system, the “Test bed for Distributed Control”, at ACIN’s Odo Struger Lab.

Bibliography

[Ald04] Aldea A., Bañares-Alcántara R., Jiménez L., Moreno A., Martínez J., and Riaño D., "The Scope of Application of Multi-Agent Systems in the Process Industry: Three Case Studies" Expert Systems with Applications Journal, January 2004, pp 39-47.

[AlS07] Al-Safi, Y. and V. Vyatkin (2007). An Ontology-Based Reconfiguration Agent for Intelligent Mechatronic Systems. In: HoloMAS 2007, LNAI 4659, pp. 114-126. Springer Berlin-Heidelberg.

[Aus62] Austin J. L.. "How to do things with words". Clarendon Press, Oxford, UK, 1962.

[Aze00] Azevedo A.L. and Sousa J.P. "A component-based approach to support order planning in a distributed manufactusring enterprise"Journal of Materials Processing Technology 107 (2000) 431±438

[Bab05] Babiceanu R. F., Chen F. F., and Sturges R. H., "Real-time holonic scheduling of material handling operations in a dynamic manufacturing environment," Robotics and Computer-Integrated Manufacturing, vol. 21, pp. 328-337, 2005.

[Bab06] Babiceanu, R. F., & Chen, F. F. (2006). Development and applications of holonic manufacturing systems: A survey. Journal of Intelligent Manufacturing, 17, 111–131.

[Bak91] Baker D. A., Factory Control in Multi-Agent Heterarcies, Journal of Manufacturing Systems May 8, 1998

[Bar04] Barata de Oliveira, J.A. (2004), "Coalition based approach for shop floor agility – a multi-agent approach", Universidade Nova de Lisboa, Lisboa, PhD thesis,

[Bar05] Barata, J.: Coalition Based Approach for Shop Floor Agility – A Multiagent approach. Edições Orion, Amadora – Lisboa (2005)

[Bas05] Bastos R., Oliveira F. and Oliveira J., Autonomic computing approach for resource allocation, Expert Systems with Applications 28 Elsevier Ltd, (2005), 9-19.

[Bed91] Bedworth, D. D., Henderson M. R., and Wolfe P. M.. Computer- Integrated Design and Manufacturing. New York: McGraw-Hill, Inc., 1991

[Bel07] Bellifemine F., Caire G. and Greenwood D., Developing Multi-agent Systems with JADE, WILEY, April 2007

[Ben06] Benjamin P., Patki M. and Mayer R.: Using ontologies for simulation modeling. Winter Simulation Conference 2006: 1151-1159

[Ber99] Bernard SM, Wang XK, Liu CY (1999) Expert, neural and fuzzy systems in process planning. Tsinghua Sci Technol No.1.

[Ber01] Berners-Lee T., Hendler J., and Lassila O., (2001). The semantic web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific american, 2001, vol. 5, n°. 284. p. 28-31.

[Ber03] Bertier, M.; Marin, O.; Sens, P., "Performance analysis of a hierarchical failure detector" Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on 22--25 June 2003 Page(s):635—644

[Bla91] Black, J T. The Design of the Factory with a Future. New York: McGraw-Hill,1991.

[Bla01] Blazewick, J.: Scheduling computer and manufacturing processes. Springer (2001)

[Boc04] Boccalatte, A., Gozzi, A., Paolucci, M., Queirolo, V., & Tamoglia, M. (2004). A multi-agent system for dynamic just-in-time manufacturing production scheduling. IEEE International Conference on Systems,Man and Cybernetics, 6, 5548–5553.

[Bon97] Bongaerts, L., Van Brussel, H., Valckenaers, P. and Peeters, P. (1997) Reactive scheduling in holonic manufacturing systems: architecture, dynamic model and cooperation strategy. In: Proceedings of the ASI-97 of NOE on ICIMS pp. 1-8, Budapest.

[Boo05] Boothroyd, G. "Assembly Automation and Product Design, 2nd Edition", Taylor and Francis, Boca Raton, Florida, 2005.

[Bos99] Bose, U. (1999) ‘A cooperative problem solving framework for computer-aided process planning’,. Proceedings of the 32nd Hawaii International Conference on System Sciences - 1999

[Bra04] Brachman R. and Levesque H., “Knowledge Representation and Reasoning” Elsevier/Morgan Kaufmann 2004

[Bray00] Bray, T., et al (eds.), “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation, http://www.w3.org/TR/2000/REC-xml-20001006, 2000.

[Bre00] Brennan R.W. and O W., “A simulation test-bed to evaluate multi-agent control of manufacturing systems,” in Proc. Winter Simulation Conference 2000, J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, Eds., Orlando, Florida, USA, 2000, pp. 1747-1756.

[Bre07] Brennan R. W., Toward Real-Time Distributed Intelligent Control: A Survey of Research Themes and Applications, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 37(5): 744-765, 2007.

[Bro86] Brooks, R.A. 1986. A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation RA-2(1): 14-23.

[Bru98] Brussel H. V., Wyns J., Valckernaers P., and Bongaerts L., “Reference architecture for holonic manufacturing systems: PROSA,” Comput. Ind., vol. 37, pp. 255–274, 1998.

[Bus01] Bussmann S. and 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 (ETFA 2001), p.481-488 (Vol.2). Antibes Juan-les-pins, France, 2001.

[Bus04] Bussmann S., Jennings N.R. and Wooldridge M. (2004). Multiagent Systems for Manufacturing Control: A design Methodology. Springer Berlin-Heidelberg.

[Byr97] Byrne M. D. Byrne and Chutima P., "Real-time operational control of an FMS with full routing flexibility," Inter-national Journal of Production Economics, vol. 51, pp. 109-113, 1997.

[Cai01] Caire G., Leal F., Chainho P., Evans R., Jorge F.G., Juan Pavon G., Kearney P., Stark J., and Massonet P. Project p907, deliverable 3: Methodology for agent-oriented software neginnering. Technical Information Final version, European Institute for Research and Strategic Studies in Telecommunications (EURESCOM), 09 2001.

[Can07] Candido G. and Barata J., “A multiagent control system for shop floor assembly,” in HoloMAS 2007, V. Mařík, V. Vyatkin, and A.W. Colombo, Eds., LNAI 4659, Springer-Verlag Berlin Heidelberg, 2007, pp. 293-302.

[Car09] Cardoso H. L., Integrating JADE and Jess, available at: http://jade.tilab.com/doc/tutorials/jade-jess/jade_jess.html (11.02.2009)

[Cha02] Chan F. T. S., Chan H. K., and H. Lau C. W., "The State of the Art in Simulation Study on FMS Scheduling: A Comprehensive Survey," The Int. J. of Ad-vanced Manufacturing Technology, vol. 19, pp. 830-849, 2002.

[Che90] Cheng, T. C. E. and Sin, C. C. S.: A state-of-the-art review of parallel machine scheduling research, European Journal of Operational Research, Vol. 47 (1990) 271-292.

[Chr03] Christensen J. H., “HMS/FB architecture and its implementation,” in Agent-Based Manufacturing: Advances in the Holonic Approach, S. M. Deen, Ed. Berlin, Germany: Springer-Verlag, 2003, pp. 53–88.

[Chr07] Christo C. and Cardeira Carlos “Trends in Intelligent Manufacturing Systems” Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE2007), Vigo (Spain), June 4-7, 2007, Page(s):3209 - 3214.

[CLI09] CLIPS: A Tool for Building Expert Systems. http://www.ghg.net/clips/WhatIsCLIPS.html (11.02.2009)

[Col06] Colombo, A.W.; Schoop, R.; Neubert, R. :"An agent-based intelligent control platform for industrial holonic manufacturing systems," IEEE Trans. on Industrial Electronics, vol. 53, no. 1, pp. 322- 337, 2006.

[Cor05] Corsar D. and Sleeman D. Reusing JessTab Rules in Protégé. In Proceedings of AI-2005, publ: Springer, pp 7-20. 2005

[DAML] DAML Webpage, http://www.daml.org/ Accessed 16.08.08.

[Dav94] Davidsson, P., (1994), "Concepts and autonomous agents", LU-CS-TR, 94-124, Department of Computer Science, Lund University.

[Del96] Delchambre, A. (1996). CAD Method for Industrial Assembly: Concurrent Design of Products, Equipment, and Control Systems, John Wiley & Sons Ltd.

[DeL01] DeLoach S., Wood M., and Sparkman C.. Multiagent systems engineering. International Journal of Software Engineering and Knowledge Engineering, 11(3):231{258, 2001.

[Dev00] Devianov B. and Toueg S.. Failure detector service for dependable computing. In Proc. of the First Int'l Conf. on Dependable Systems and Networks, pages 14–15, juin 2000.

[Eri02] Eriksson H. The JESS TAB Approach to Protégé and JESS Integration. In Proceedings of the IFIP 17th World Computer Congress - TC12 Stream on Intelligent Information Processing, pages 237–248. Kluwer, B.V., 2002.

[Eri03] Eriksson, H.: Using JessTab to Integrate Protege and Jess. IEEE Intelligent Systems (2003), 18(2):43-50.

[Fac06] Faci N., Guessoum Z. and Marin O., "DimaX: A Fault Tolerant Multi-agent platform", In: Proceedings of the SELMAS'06., Shanghai, China (2006)

[Fav04] Favre-Bulle B.: "Automatisierung komplexer Industrieprozesse"; Springer-Verlag, Wien - New York, Wien, 2004

[Fen04] Feng, S.C.; Stouffer, K.A.; Jurrens, K.K. "Intelligent agents-enabled integrated manufacturing planning and control" The 8th International Conference on Computer Supported Cooperative Work in Design, 2004.

[Fin94a] Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML: as an Agent Communication Language. Association of Computing Machinery (1994)

[Fin94b] Finin T, Fritzson R, McEntire R (1994) KQML as an agent communication language. Proc 3rd International Conference on Information and Knowledge Management. CIKM'94), ACM

[Fin97] Finin T., Labrou Y. and Mayfield J., KQML as an agent communication language I. In: J. Bradshaw, Editor, Software Agents, MIT Press, Cambridge, MA (1997).

[Fou03] Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification http://www.fipa.org/specs/fipa00037/ and FIPA ACL Message Structure Specification http://www.fipa.org/specs/fipa00061/, (2003)

[Fox98] Fox, M., Barbuceanu, M., Gruninger M,, and Lin J. "An Organization Ontology for Enterprise Modelling" Simulating Organizations: Computational Models of Institutions and Groups, M. Prietula, K. Carley & L. Gasser (Eds), Menlo Park CA: AAAI/MIT Press, pp. 131-152

[Giu02] Giunchiglia F., Mylopoulos J., and Perini A. The Tropos software development methodology: Processes, Models and Diagrams. In Third International Workshop on Agent-Oriented Software Engineering, July 2002.

[Gol98] Goldsmith, S. Y. and Interrante, L. D. (1998) An autonomous manufacturing collective for job shop scheduling. In: The Proceedings of AI & Manufacturing Research Planning Workshop, pp. 69-74, Albuquere, AAAI Press.

[Gon06] Gonzalez E. J., Hamilton A. F., Moreno L., Marichal R. L. and Munoz V. "Software experience when using ontologies in a multi-agent system for automated planning and scheduling" Softw. Pract. Exper. 2006; (Published online)

[Gru93] Gruber T. R. „A translation approach to portable Ontologies" Knowledge Acquisition, 1993

[Gru05] Grüninger, M. and J. B. Kopena (2005), Planning and the Process Specification Language, In: Proceedings of WS2 ICAPS 2005, pp. 22-29.

[Gue04] Guessoum Z., Ziane M., Faci N., Monitoring and Organizational-Level Adaptation of Multi-Agent Systems, AAMAS'04, ACM, pp. 514-522, New York City, July 2004.

[Gun98] Gunasekaran A. "Agile Manufacturing: Enablers and an Implementation Framework," International Journal of Production Research, vol. 36, no. 5, pp. 1223-1247, May 1998.

[Gun04] Gungui, I., 2004. Integrazione di agenti logici in DCaseLP. Master's thesis, Computer Science Department of Genova University, Italy

[Gun05] Gungui, I. : *Integrating Logical Agents Into DCaseLP*, MSc Thesis - the Department of Computer Science -. University of Genova, 2005

[Hag96] Hagg, S. (1996). A Sentinel Approach to Fault Handling in Multi-Agent Systems. Proceedings of the Second Australian Workshop on Distributed AI, in conjunction with Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96), Cairns, Australia.

[Hah94] S. Hahndel and P. Levi, "A distributed task planning method for autonomous agents in a FMS," in Intelli-gent Robots and Systems '94. 'Advanced Robotic Sys-tems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on, 1994, pp. 1285-1292 vol.2.

[Har73] Harrington J. Computer integrated manufacturing. New York: Industrial Press, 1973.

[Heg08] Hegny I., Hummer-Koppendorfer O., Zoitl A., Koppensteiner G. and Merdan M., "Integrating Software Agents and IEC 61499 Realtime Control for Reconfigurable Distributed Manufacturing Systems", IEEE 3rd International Symposium on Industrial Embedded Systems - SIES 2008, France;

[Hei00] Heilala J, & Volvo P, "Modular Reconfigurable Flexible Final Assembly Systems In Electronic Industry", Assembly Automation Workshop, Netherlands, 11'-12" May 2000

[Hei01] Heilala, J. and Voho, P. (2001) 'Modular reconfigurable flexible final assembly systems', Assembly. Automation, Vol. 21, No. 1, pp.20–28.

[Hel04] Helsinger, A., Thome, M., and Wright, T. "Cougaar: A Scalable, Distributed Multi-Agent Architecture." In Proceedings of the 2004 IEEE Conference on Systems, Man, and Cybernetics. The Hague, The Netherlands. October, 2004.

[Hev01] Heverhagen T. and Tracht R., "Integrating UML-Real Time and IEC 61131-3 with function block adapters," in Proc. 4th IEEE Int. Symp. Object-Oriented Real-Time Distrib. Comput. (ISORC 2001), May 2–4, pp. 395–402.

[Hil98] Hiller M., Software Fault-Tolerance Techniques from a Real-Time Systems Point of View. Department of Computer Engineering, Chalmers University of Technology, Göteborg, 1998.

[Hob87] Hobbs, J., Croft W., Davies T., Edwards D., and Laws K., (1987). The TACITUS Commonsense Knowledge Base, Artificial Intelligence Research Center, SRI International.

[Hod05] Hodík, J. Bečvář, P. Pěchouček, M. Vokřínek, J. and Pospíšil, J.: ExPlanTech and ExtraPlanT: multi-agent technology for production planning, simulation and extra-enterprise collaboration. International Journal of Computer Systems Science and Engineering. 2005, vol. 20, p. 357-367

[Hol95] Hollis, R. and Quaid, A.: An Architecture for Agile Assembly American Society of Precision Engineering 10th Annual Mtg, October, 1995.

[Hol97] Holthaus, O. 1997. "Design of Efficient Job Shop Scheduling Rules". Computers and Industrial Engineering 33, No. 1, 245-252.

[Hol06] Holmström P., "Modelling Manufacturing Systems Capability", Production Engineering, Dissertation. 2006-06-07, TRITA-IIP-06-04.

[Huh02] Huhns, MN, Holderfield, VT, and Gutierrez, RLZ Achieving Software Robustness via Large-Scale Multiagent Systems. In Proceedings of SELMAS. 2002, 199-215.

[Hur01] Hurink J. and Knust S., "Makespan minimization for flow-shop problems with transportation times and a single robot," Discrete Applied Mathematics, vol. 112, pp. 199-216, 2001.

[IEC04] IEC TC65/WG6, IEC 61499: Function blocks for industrial-process measurement and control systems – Parts 1 to 4. Geneva: International Electrotechnical Commission (IEC), 2004-2005.

[IEC31] IEC TC65/WG6, Programmable controllers – Part 3: Programming languages. Geneva: International Electrotechnical Commission (IEC),1993.

[IEC99] IEC TC65/WG6, IEC 61499: Function blocks for industrial-process measurement and control systems – Parts 1 to 4. Geneva: International Electrotechnical Commission (IEC), 2004-2005.

[JAD08] JADE - Java Agent Development Framework, http://jade.tilab.com/, Accessed July 2008.

[Jen98] Jennings N. R. and Wooldridge M., "Applications of intelligent agents," in Agent Technology, N. R. Jennings and M. J. Wooldridge, Eds: Springer, 1998, pp. 3–28.

[Jen01] Jennings N.R., An agent-based approach for building complex software systems, Communications of the ACM 44 (4) (2001) 35– 41.

[Jen03] Jennings, N.R., Bussman, S. Agent-Based Control Systems: Why are They Suited to Engineering Complex Systems? IEEE Control Systems Magazine, 23(3), 2003, pp. 61-73.

[Joh01] John, K. and Tiegelkamp, M. (2001) IEC 61131-3: Programming Industrial Automation Systems, Springer.

[Jon86] Jones, A. T. and McLean, C. R.: "A proposed hierarchical control model for automated manufacturing systems," J. Manufac. Syst., vol. 5, no. 1, pp. 15–25, 1986.

[Kal07] Kalogeras, A.P.; Ferrarini, L.; Lueder, A.; Gialelis, J.; Alexakos, C.; Peschke, J.; Veber, C.;, "Ontology-driven control application design methodology", ETFA07, IEEE Conference on Emerging Technologies & Factory Automation, 25-28 Sept. 2007, P. 1425 - 1428.

[Kam98] Kaminka G. A. and Tambe M.. What is Wrong With Us? Improving Robustness Through Social Diagnosis. In Proceedings of the 15 th National Conference on Artificial Intelligence (AAAI-98), 1998.

[Kha05] Khan Z.A., Shahid S., Ahmad H.F., Ali A., and Suguri H., "Decentralized architecture for fault tolerant multi agent system," Proceedings of Autonomous Decentralized Systems, pp. 167–174, April 2005, Chengdu, Jiuzhaigou, China.

[Kid94] Kidd, P. T. (1994). Agile manufacturing: Forging new frontiers. England: Addition-Wesley.

[Kim06] Kim K. Y,. Manley D. G., Yang and H. J., "Ontology-based Assembly Design and Information Sharing for Collaborative Product Development," Computer-Aided Design (CAD), Vol. 38, 2006, pp. 1233-1250.

[Kis06] Kishore R., Zhang H. and Ramesh R. "Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems" Decision Support Systems 42 (2006) 48– 78

[Kop08a] Koppensteiner G.; Merdan M.; Zoitl A. & Favre-Bulle B. (2008): Ontology-based Resource Allocation in Distributed Systems using Director Facilitator Agents, IEEE International Symposium on Industrial Electronics ISIE'08, Cambridge, UK

[Kop08b] Koppensteiner G., Merdan M., Hegny I., and Weidenhausen G. (2008): A Change-Direction-Algorithm for distributed Multi-Agent Transport Systems, 2008 IEEE International Conference on Mechatronics and Automation, Takamatsu, Japan, 2008.

[Kop08f] Koppensteiner G., Merdan M., Zoitl A., Favre-Bulle B.(2008): Ontology-based Resource Allocation in Distributed Systems using Director Facilitator Agents, IEEE International Symposium on Industrial Electronics, Cambridge, United Kingdom,

[Kop09] Koppensteiner G., Merdan M., Lepuschitz W., Hegny I., Hybride Based Approach for Fault Tolrence in a Multi-Agent System, IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM'2009), Singapore,2009 (submitted)

[Kro99] Krothapalli, N. K. C. and Deshmukh, A. V. (1999) Design of negotiation protocols for multi-agent manufacturing systems. International Journal of Production Research, 37 (7), 1601-1624.

[Kul05] Kulvatunyou B., H. Cho, and Y. Son, "A semantic web service framework to support intelligent distributed manufacturing," International Journal of Knowledge-Based and Intelligent Engineering Systems, Vol. 9, No. 2, pp. 107-127, 2005.

[Kum00] Kumar, S., Cohen, P.: Towards a fault-tolerant multi-agent system architecture. In: Proceedings of the Fourth International Conference on Autonomous Agents, ACM Press New York, NY, USA (2000)

[Kum06] Kumar M. and Rajotia S. "Integration of process planning and scheduling in a job shop environment" Int J Adv Manuf Technol (2006) 28: 109–116

[Kut95] Kutanoglu, E. and Sabuncuoglu, I., 1995, An investigation of reactive scheduling policies under machine breakdowns. Proceedings of the 4th Industrial Engineering Research Conference, pp. 904± 913.

[Kut01] Kutanoglu E. and Sabuncuoglu I., "Routing-based Reactive Scheduling Policies for Machine Failures in Job Shops," Int J of Production Research, vol. 39, pp. 3141-3158 (2001)

[Lab01] Labrou Y., Standardizing agent communication. In: V. Marik and O. Stepankova, Editors, Multi-Agent Systems and Applications (Advanced Course on Artificial Intelligence) (2001), pp. 74–97

[Las06] Lastra, J. L. M, and Delamer I M. "Semantic Web Services in Factory Automation:Fundamental Insights and Research Roadmap" IEEE Transactions On Industrial Informatics, VOL. 2, NO. 1, February 2006 1

[Lee01] Lee, C. -Y. and Z. L. Chen,, "Machine scheduling with transportation considerations," Journal of Scheduling, vol. 4, pp. 3-24, 2001.

[Lee06] Lee C. Y., Leung J. Y. T., and Yu G., "Two Machine Scheduling under Disruptions with Transportation Considerations," Journal of Scheduling, vol. 9, pp. 35-48, 2006.

[Lei04] Leitão P., An Agile and Adaptive Holonic Architecture for Manufacturing Control, doctoral dissertation, Dept. Electrical and Computer Eng., Univ. of Porto, Portugal, 2004.

[Lem06] Lemaignan, S. Siadat, A. Dantan, J.-Y. Semenenko, A. „MASON: A Proposal For An Ontology Of Manufacturing Domain" IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06) pp. 195-200

[Lim04] Lim M.K. and Zhang D.Z. "An integrated agent-based approach for responsive control of manufacturing resources" Computers & Industrial Engineering 46 (2004) 221–232

[Loh05] Lohse N., Hirani, H. and Ratchev, S., 2005. An Ontology for the Definition and Validation of Assembly Processes for Evolvable Assembly Systems. *In:* 6th IEEE International Symposium on Assembly and Task Planning, Montreal, Canada.

[Lop06] Lopez, O. and J.L.M. Lastra (2006). Using Semantic Web Technologies to Describe Automation Objects. Int. J. Manufacturing Research, 1(4):482-503.

[Lop07] Lopez O. J. and Lastra J. M.. A Real-Time Interface for Agent-Based Control. IEEE Second International Symposium on Industrial Embedded Systems (SIES 07). Lisboa, Portugal. July 2007.

[Mal07] Malec J., Nilsson A., Nilsson K. and Nowaczyk S. “Knowledge-Based Reconfiguration of Automation Systems” IEEE Conference on Automation Science and Engineering, Scottsdale, AZ, USA, Sept 22-25, 2007

[Mar05] Martinez Lastra J.L., Torres E.L. and Colombo A.W. “A 3D visualization and simulation framework for intelligent physical agents,“. In HoloMAS 2005, V. Mařík, R. Brennan, and M. Pěchouček, Eds., LNAI 3593, Springer-Verlag Berlin Heidelberg, 2005, pp. 23-38.

[Mar05a] Marık, V. Vrba, P. Hall, K.H. Maturana, F.P.: Rockwell Automation Agents for Manufacturing. In The Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems – Special Track for Industrial Applications. New York: ACM, 2005, s. 107-113.

[Mat96] Maturana, F. and Norrie, D. (1996). Multi-Agent Mediator Architecture for Distributed manufacturing. Journal of Intelligent Manufacturing, 7:257-270.

[Mat04] Maturana F.P., Tich P., Slechta P., Discenzo F., Staron R.J. and Hall K., Distributed multi-agent architecture for automation systems, Expert Systems with Applications 26 (2004) (1), pp. 49–56

[McL05] McLean, C., Lee Y. T., Shao G. and Riddick F. 2005. Shop Data Model and Interface Specification, NISTIR 7198. National Institute of Standards and Technology, Gaithersburg, MD.

[Mel05] Mellouli, S., FATMAS: a methodology to design fault-tolerant multi-agent systems. Ph.D. Thèse, Université Laval, 2005

[Mer08] Merdan M., Koppensteiner G., Hegny I., Favre-Bulle B.: "Application of an Ontology in a Transport Domain"; IEEE International Conference on Industrial Technology, Sichuan University, Chengdu, China; 2008

[Mer08c] Merdan M.; Koppensteiner G.; Zoitl A. & Hegny I., (2008) Intelligent-Agent based Approach for Assembly Automation, IEEE Conference on Soft Computing in Industrial Applications, June 2008, SMCia/08 Muroran, Japan

[Mer08d] Merdan M., Moser T, Wahyudin D., Biffl S., and Vrba P. "Simulation of Workflow Scheduling Strategies Using the MAST Test Management System"; 10th International Conference on Control, Automation, Robotics and Vision (ICARCV), Hanoi, Vietnam; 2008.

[Mer08e] Merdan M., Vittori L., Koppensteiner G., Vrba P., and Favre-Bulle B. "Simulation of an ontology-based multi-agent transport system", International Conference on Instrumentation, Control and Information Technology (SICE), Tokyo, Japan, 2008.

[Mer09] Merdan M. Lepuschitz W. Hegny I. and Koppensteiner G. „Application of a Communication Interface between Agents and the Low Level Control” 4th International Conference on Autonomous Robots and Agents. Wellington, New Zealand, 2009.

[Mil05] Miller, J. and Baramidze G.. (2005). Simulation and the semantic web. Proceedings of the 2005 Winter Simulation Conference. Piscataway, New Jersey:Institute for Electrical and Electronics Engineers.

[Mok01] Mokotoff, E. (2001) ‘Parallel machine scheduling problems: a survey’, Asia-Pacific Journal of Operational Research, Vol. 18, No. 2, pp.193–242.

[Mon03] Monch L., Stehli M. and Zimmermann J., “FABMAS: an agentbased system for production control of semiconductor manufacturing process,” in HoloMAS 2003, V. Mařík, D. McFarlane, and P. Valckenaers, Eds., LNAI 2744, Springer-Verlag Berlin Heidelberg, 2003, pp. 258-267.

[Mon05] Mönch L., Stehli M., „ManufAg: a multi-agent-system framework for production control of complex manufacturing systems” Springer-Verlag 2005

[Mor05] Morel, G., Valckenaers, P., Faure, J. M., Pereira, C., & Diedrich, C. (2005). Manufacturing plant control: Challenges and open issues. In Proceedings of the 16th IFAC Triennial World Congress

[Mos05] Mostefai S., Bouras A. and Batouche M. “Effective Collaboration in Product Development via a Common Sharable Ontology” International Journal of Computational Intelligence Volume 2 Number 4 2005

[Nec91] Neches, R., Fikes R., Finin T., Gruber T., Patil R., Senator T., and Swartout W. R. (1991). Enabling technology for knowledge sharing. AI Magazine 12(3):36-56.

[Obi02] Obitko, M. and V. Mařík (2002). Ontologies for Multi-Agent Systems in Manufacturing Domain, Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA’02)

[Obi08] Obitko M., Vrba P., Marik V. and Radakovic M. “Semantics in Industrial Distributed Systems “The 17th IFAC World Congress, Seoul, Korea, 2008

[Oki93] Okino N., Bionic Manufacturing System- Flexible Manufacturing Systems Past-Present-Future, 1993, pages 73–95.

[Oli94] Oliveira E., “Cooperative multi-agent system for an assembly robotics cell,” Robot. Comput. Integr. Manufac., vol. 11, no. 4, pp. 311–317, 1994.

[Oue98] Ouelhadj D., Hanachi C., and Bouzouia B., “Multi-agent system for dynamic scheduling and control in manufacturing cells,” in Working Notes of the Agent-Based Manufacturing Workshop, Minneapolis, MN, 1998, pp. 96–105.

[Oue07] Ouelhadj, D. & Petrovic, S. Survey of Dynamic Scheduling in Manufacturing Systems, Journal of Scheduling, t/a: t/a, 2007 (accepted)

[OWL] Web Ontology Language OWL, http://www.w3.org Accessed 16.08.08.

[Par01] Parunak F.V.D., Baker A.D., and Clark S.J., "The AARIA agent architecture: From manufacturing requirements to agent-based system design", Integrated Conipuler-Aided Engineering, Vol. 8, No. 1, pp. 45-58, 2001.

[Par87] Parunak, V. 1987. Manufacturing Experience with the Contract Net. In Distributed Artificial Intelligence, Volume 1, ed. M. Huhns, 285–310. London: Pitman.

[Par96] Parunak, H. V. (1996). Applications of distributed artificial Intelligence in industry. In: Foundation of Distributed Artificial Intelligence, Chapter 4, Eds. O’Hare, G. M. P. and Jennings, N. R., Wiley Interscience, New York.

[Par97] Parunak, H.V.D. and VanderBok, R.: Managing Emergent Behavior in Distributed Control Systems. In Proceedings of IAS-TECH/97 conference, Anaheim, CA, 1997

[Par98] Parunak H.V.D., Baker A. and Clark S., The AARIA agent architecture: From manufacturing requirements to agent-based system design, in: Working Notes of the ABM Workshop, Minneapolis, MN, 1998, pp. 136–145.

[Pat05] Patil L., Dutta D., Sriram R.D., Ontology-based exchange of product data semantics, IEEE Transactions on Automation Science and Engineering 2 (3) (2005) 213-225.

[Pec02] Pechouček M., Rıha A., Vokrınek J., Marık V., and Prazma V., “Explantech: applying multi-agent syssems in production planning,” International Journal of Production Research, vol. 40, no. 15, pp. 3681–3692, 2002.

[Pec05] Pechoucek, M., Rehak, M. and Marik, V. "Expectations and Deployment of Agent Technology in Manufacturing and Defence: Case Studies," Proc. 4th Int'l Conf. Autonomous Agents and Multi-Agent Systems—AAMAS 2005 Industry Track, ACM Press,2005.

[Pec07] Pechouček, M., Rehák, M., Charvát, P., Vlček, T. and Kolář, M. (2007): Multi-Agent Planning in Mass-Oriented Production. IEEE Trans. SMC, part C, vol. 37, No.3, pp. 386-395

[Pec08] Pechoucek M. and Marik V.: Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies. International Journal on Autonomous Agents and Multi-Agent Systems. 2008, DOI10.1007/s10458-008-9050-0

[Pen99] Peng Y., Finin T., Labrou Y., R. Cost S., Chu B., Long J., Tolone W. J., and Boughannam A., “Agent-based approach for manufacturing integration: The CIIMPLEX experience,” Appl. Artif. Intell., vol. 13, no. 1, pp. 39–63, 1999.

[Pes01] Pesenti, R., Castelli, L., Santin, P., 2001, Scheduling in einer realistischen Umgebung Mit Autonome Agenten: A Simulation Study. Proceedings of Agent Based Simulation II Workshop, Passau, Germany, April 2-4, pp.149-154. Proceedings of Agent-Based-Simulation-II Workshop, Passau, Deutschland, April 2-4, pp.149-154.

[Pin02] Pinedo, M., Scheduling theory, algorithms and systems, Second edition, Prentice Hall, 2002.

[Pla08] Platon E., Sabouret N., Honiden S., An architecture for exception management in multiagent systems, International Journal of Agent-Oriented Software Engineering, v.2 n.3, p.267-289, 2008

[Qiu05] Qiu X.: Agent Interaction in a Semantic Web Environment: A state-of-the-art survey and prospects in knowledge mobilization. Information Systems Research in Scandinavia, IRIS'28, Kristiansand, Norway, 2005.

[Rab93] Rabemanantsoa, M., Pierre, S., "Knowledge-based System for Assembly Process Planning", in Proceedings of IEEE Software Engineering Standards Symposium (SESS93), Brighton, England, September 1- 3, 1993, pp. 267-272.

[Raj99] Rajendran C. and Holthaus O., "A comparative study of dispatching rules in dynamic flowshops and job-shops," European Journal of Operational Research, vol. 116, pp. 156-170, 1999.

[Raj04] Rajakumar S., Arunachalam V. P., and Selladurai V., "Workflow balancing strategies in parallel machine scheduling," The International Journal of Advanced Manufacturing Technology, vol. 23, pp. 366-374, 2004.

[Raj06] Rajpathak, D.G. and Motta, E. and Zdrahal, Z. and Roy, R. (2006) A generic library of problem solving methods for scheduling applications. IEEE Transactions on Knowledge and Data Engineering, 18 (6). pp. 815-828.

[RDF] Resource Description Framework RDF, http://www.w3.org/RDF/ Accessed 16.08.08.

[RDFS] RDF Schema http://www.w3.org/TR/rdf-schema Accessed 16.08.08.

[Saa97] Saad, A., Kawamura, K., and Biswas, G., "Performance Evaluation of Contract Net-Based Heterarchical Scheduling for Flexible Manufacturing Systems", Special Issues on Intelligent Manufacturing Planning and Shop Floor Control, International Journal of Intelligent Automation and Soft Computing, 1997.

[Sab00] Sabuncuoglu, I. and Bayiz, M., Analysis of reactive scheduling problems in a job shop environment, European Journal of Operational Research, 126 (3), 567-586 (2000).

[Sandi07] Sandia National Laboratories, Jess: the Rule Engine for the JavaTM Platform. Available at: http://herzberg.ca.sandia.gov/, last visited December 2007.

[Sch00] Schoop R. and Neubert R., Agent-Oriented Material Flow Control System Based on DCOM, Proceedings of the Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, p.342, March 15-17, 2000

[Sei03] Seilonen, I., Pirttioja, T., Appelqvist, P., Halme, A., Koskinen, K.: Distributed Planning Agents for Intelligent Process Automation, Accepted to the 5th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2003). Kobe, Japan (2003)

[She98] Shen W., and Norrie H.N., "An agent-based approach for dynamic manufacturing Scheduling", Proceedings of Autonomous Agents '98 Workshop on Agent-Based Manufacturing, Minneapolis/St.Paul, MN, pp. 117-128,1998.

[She99] Shen W., Norrie D.H.: Agent-based Systems for Intelligent Manufacturing : A State-of-the-Art Survey, Knowledge and Information Systems, an International Journal, Vol.1, No.2, 1999, p. 129-156.

[She00a] Shen W., Maturana F. and Norrie D.H., MetaMorph II: an agent-based architecture for distributed intelligent design and manufacturing, Journal of Intelligent Manufacturing 11 (3) (2000) 237–251.

[She00] Shen, W., Barthes J. P., and Norrie D. H.,. Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing, Taylor & Francis, 2000.

[She01] Shen, W, Norrie, DH, and Barthes, JA, Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing, Taylor & Francis, London, 2001.

[She02] Shen W.M., Distributed manufacturing scheduling using intelligent agents, IEEE Intell Syst Appl 17 (2002)

[She06] Shen W.M., Wang L.H. and Hao Q., Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey, IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews 36 (4) (2006), pp. 563–577.

[She07] Shen W. & Norrie D.(1997), " Facilitator, Mediator or Autonomous Agents", In Proceedings of the Second International Workshop on CSCW in Design, Bangkok, Thailand. pp. 119--124.

[Smi80] Smith R (1980) The contract net protocol: high level communication and control in distributed problem solver. IEEE Transactions on Computers, 29:1104-1113.

[Smi97] Smith S. and Becker M., An Ontology for Constructing Scheduling Systems; Working Notes of 1997 AAAI Symposium on Ontological Engineering, AAAI Press, March, 1997

[Sny04] Snyder, R., Tomlinson, R.T.: Robustness Infrastructure for Multi-Agent Systems. In: Proceedings of the Open Cougaar, N.Y., U.S.A (2004)

[Stanf07] Stanford Medical Informatics, Stanford University. Protege Website. http:// protege.stanford.edu., Accessed December 2007.

[Sto00] Stone P. and Veloso M., Multiagent systems: a survey from a machine learning perspective, Autonomous Robots, volume 8, number 3. 2000.

[Sun01] Sun, J. and Xue, D. (2001) A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources. Computers in Industry, 46 (2), 189-207.

[Sun05] Sunder C., Zoitl A., Strasser T., and Favre-Bulle B., “Intuitive control engineering for mechatronic components in distributed automation systems based on the reference model of IEC 61499,” in 3rd IEEE International Conference on Industrial Informatics, 2005 (INDIN ’05). Proceedings of, August 2005, pp. 50–55.

[Syc98] Sycara K.. Multiagent systems. AI Magazine, 19(2):79- 92, 1998.

[Tay11] Taylor, F.W. (1911). The principles of scientific management. New York: Harper.

[Teh08] Tehrani H., Sugimura N., Iwamura K. And Tanimizu Y., “Integrated Dynamic Process Planning and Scheduling in Flexible Manufacturing Systems via Autonomous Agents”, Journal of Advanced Mechanical Design, Systems, and Manufacturing, Vol. 2, No. 4 (2008), pp.719-734 .

[Tha96] Tharumarajah A., Wells A. J., and Nemes L., “Comparison of the bionic, fractal and holonic manufacturing system concepts,” Int. J. Comput. Integr. Manuf., vol. 9, no. 3, pp. 217–226, May 1996.

[Tha04] Thadakamalla H. P., Raghavan U. N., Kumara S., and Albert R., "Survivability of Multiagent-Based Supply Networks: A Topological Perspective," IEEE Intelligent Systems Magazine, vol. 19, iss. 5, pp. 24— 31, 2004.

[Thr05] Thramboulidis, K., “IEC 61499 in Factory Automation”, Int. Conf. on Industrial Electronics, Technology & Automation (CISSE-IETA 05), Dec. 10-20, 2005

[Tic06] Tichy P., Slechta P., Staron R. J., Maturana F. P., and Hall K. H., “Multiagent Technology for Fault Tolerance and Flexible Control,” IEEE Transactions On Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 36, iss. 5, pp. 700-704, 2006.

[Tom05] Tomás V. R., and Garcia L. A., “A Cooperative Multiagent System for Traffic Management and Control”, Proc. of AAMAS’05, Utrecht, The Netherlands, July 25-29, 2005.

[UML] Unified Modeling Language UML, http://www.uml.org/ Accessed 16.08.08.

[Usc98] Uschold M., King M., Moralee S. and. Zorgios Y (1998) The Enterprise Ontology The Knowledge Engineering Review , Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate).

[Ush03] Usher J. M., “Negotiation-based routing in job shops via collaborative agents,” J. Intell. Manuf., vol. 14, no. 5, pp. 485–499, 2003.

[Utp99] Utpal B (1999) A cooperative problem solving framework for computer-aided process planning. Proc 32nd Hawaii International Conference on System Sciences

[Val04] Valckenaers P: “Challenges of Next Generation Manufacturing Systems” in Integration of Software Specification Techniques forApplications in Engineering: H. Ehrig et al. (Eds.):pp. 23–28, 2004.

[Vie03] Vieira, G. E., Hermann, J. W., and Lin, E., Re-scheduling manufacturing systems: a framework of strategies, policies and methods, Journal of Scheduling, 6 (1), 36-92 (2003).

[Vis98] NRC. (1998). Visionary Manufacturing Challenges for 2020. Washington: National Academy Press

[Vos01] Vos J.A.W.M. “ Module and System Design in Flexibly Automated Assembly”, Ph.D. thesis, Delft University of Technology, Delft, The Netherlands, 2001.

[Vrb05] Vrba, P., and Mařík, V. (2005): From Holonic Control to Virtual Enterprises: The Multi-Agent Approach. In: The Industrial Information Technology - Handbook. Boca Raton: CRC Press, pp. 107-1-107-19.

[Vrb08] Vrba P., Marík V. & Merdan M.: Physical Deployment of Agent-based Industrial Control Solutions: MAST Story: 2008 IEEE International Conference on Distributed Human-Machine Systems Athens, Greece MARCH 9-12, 2008,

[Vya02] Vyatkin V., Hanisch H.-M., and Karras S., IEC 61499 as an architectural framework to integrate formal models and methods in practical control engineering, Congress Electric Automation SPS/IPC/Drives, Nuernberg, Germany, November 2002.

[Vya05] Vyatkin, V., J. Christensen, J.L.M. Lastra and F. Auinger (2005). OOONEIDA: An Open, Object-Oriented Knowledge Economy for Intelligent Industrial Automation. IEEE Transactions on Industrial Informatics, 1(1):4-17.

[W3CSe] W3CSemanticWeb. W3C Semantic Web. http://www.w3.org/ 2001/sw/ Accessed 16.08.08.

[Wan98] Wang L., Balasubramanian S. and Norrie DH. “Agent-based Intelligent Control System Design for Real-time Distributed Manufacturing Environments,” Notes of Autonomous Agents'98 Workshop on Agent-Based Manufacturing, Minneapolis, 1998, pp. 152-159.

[War93] Warnecke H. J., The Fractal Company, Springer, Berlin, 1993.

[Wass99] Wasson, G.: Design of Representation Systems for Autonomous Agents, University of. Virginia, Dissertation, 1999.

[Wel95] Welmann, M. P. (1995). Market-oriented programming: Some early lessons. In S. H. Clearwater (Ed.), Market-based control: A paradigm for distributed resource allocation. Singapore: World Scientific.

[Whi96] Whitney, D., The Potential for Assembly Modeling in Product Development and Manufacturing. MIT Press, 1996

[Wom90] Womack J. P., Jones D. T., and Roos D., The Machine that Changed the World. New York: Simon & Schuster, Inc., 1990.

[Won06] Wong T. N., Leung C. W., Mak K. L. and Fung R. Y. K., "Dynamic shopfloor scheduling in multi-agent manufacturing systems", Expert Systems with Applications, Vol 31, No.3, 2006, pp 486-494.

[Woo86] Woods W. A., "*Important issues in knowledge representation*,"Proc. IEEE, vol . 74, no. 10, pp. 1322-1334, Oct. 1986.

[Woo98] Wooldridge, M., Jennings, N.R.: Pitfalls of agent-oriented development. In: AGENTS ’98: Proceedings of the second international conference on Autonomous agents, ACM Press (1998) 385–391

[Woo99] Wooldridge M. Intelligent Agents. In Weiss Gerhard editor .). Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge, MA, London, 1999. pp. 27-78

[Woo00] Wooldridge M., Jennings N.R., and Kinny D.. The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems, 3(3), 2000.

[Wu07] Wu J. and Durfee E. H.. Sequential resource allocation in multi-agent systems with uncertainties. In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, 2007.

[Xua04] Xuan P. A Mechanism for Robust Plan Adaptation in Large Scale Multiagent Operations. In Proceedings of the Workshop on Challenges in the Coordination of Large Scale Multiagent Systems, AAMAS 2004.

[Yin07] Yingzi W., Kanfeng G., Hongjun L. and Donglai L., (2007) Contract Net based Scheduling Approach using Interactive Bidding for Dynamic Job Shop Scheduling, IEEE International Conference on Integration Technology, , Shenzhen, China, pp: 281-286

[Zei08] Zeichen G., Zoitl A., Prenninger J.: "Systemtechnologien für Produktionssysteme"; ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb, 103 (2008), 9; S. 589 - 593.

[Zha00] Zhao, F.L., Tso, S.K., and Wu, P.S.Y, "A cooperative agent modelling approach for process planning", 2000, Computers in Industry, Vol.41, 83-97;

[Zha03] Zha, X.F., Lim, S.Y.E. and W. F. Lu, “A Knowledge Intensive Multi-Agent System for Cooperative/Collaborative Assembly Modeling and Process Planning”Journal of Integrated Design and Process Science MARCH 2003, Vol. 7, No. 1, pp. 100-121

[Zha07] Zhang W. J. and Xie S. Q. “Agent technology for collaborative process planning: a review” Int. J. Adv. Manuf.Technol, (2007) 32: 315–325

[Zhu00] Zhu D., Zhao L., Zhang J.. “A Method of Generating Assembly Plan Based on Level Hierarchy Connection Relation Model”, Journal of East China Shipbuilding Institute.14(1): 71-75, Jan 2000.

[Zoi06a] Zoitl A., Smodic R., and Grabmair G., “Enhanced real-time execution of modular control software based on IEC 61499,” in Proc. IEEE Int. Conf. Robot. Autom., May 2006, pp. 327–332.

[Zoi06b] Zoitl A., Sünder C.K., Terzic I., "Dynamic Reconfiguration of Distrubuted Control Applications with Reconfiguration Services based on IEC 61499," in 2006 IEEE Workshop on Distributed Intelligent Systems, IEEE Computer Society, June 2006, pp.. 109 - 114.

[Zoi07] Zoitl A., Strasser T., Hall K., Staron R., Sünder C.K., Favre-Bulle B., "The Past, Present, and Future of IEC 61499," in Holonic and Multi-Agent Systems for Manufacturing", Sep. 2007, pp. 1 - 14.

[ZoiDiss07] Zoitl A."Basic Real-Time Reconfiguration Services for Zero Down-Time Automation Systems"; Institut für Automatisierungs- und Regelungstechnik, TU Vienna, Austria 2007

A. Rules

In order to explain the rule-based behavior of our agents, in this section we will present a few simplified rules. Since the whole system has more than 200 of them, all other rules as well as the source code are available from the author upon request.

After the agent of conveyor CC1 gets a message that the specific pallet enters it, the rule *CA_add_newPallet* will fire causing that this conveyor updates its knowledge base by adding the a new pallet in its *hasPallet* multislot (Figure A.1). The information about the pallet is included in the content of the message.

```
(defrule CA_add_newPallet
    (SystemTime
        (time ?time))
    (object (is-a Machine_Agent)
        (OBJECT ?ca)
        (receive $? ?msg $?)
        (work_as ?conveyor)
        (:NAME ?name))
   (object (is-a Conveyor)
        (OBJECT ?conveyor)
        (:NAME ?conname))
    (object (is-a Message)
        (OBJECT ?msg)
        (conversation-id ?param)
        (performative "SUBSCRIBE")
        (content ?palletXML))
   =>
    (purge ?msg)
; purge function removes an object from Jess's Working Memory and form Protege
    (bind $?allPallets (slot-get ?conveyor hasPallet))
    (bind ?pallet (from-xml-pallet ?palletXML))
; for given string containing xml file function from-xml-pallet creates pallet instance
    (slot-insert$ ?conveyor hasPallet (+ (length$ (slot-get ?ca hasPallet)) 1) ?pallet))
```

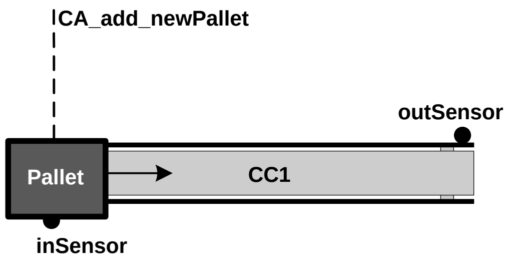
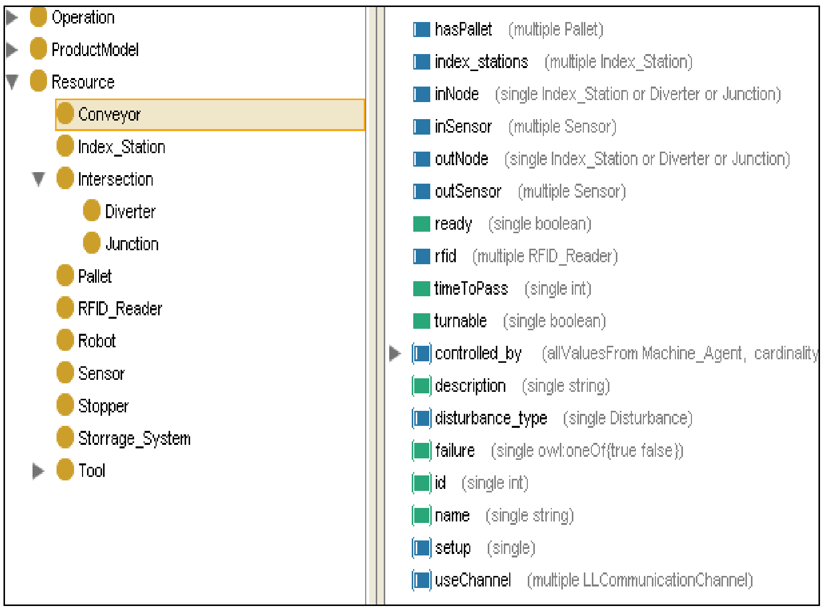


Figure A.1: Conveyor receives a new pallet

A part of the conveyor ontology is presented in Figure A.2.



- Operation
- ProductModel
- Resource
  - Conveyor
  - Index_Station
  - Intersection
    - Diverter
    - Junction
  - Pallet
  - RFID_Reader
  - Robot
  - Sensor
  - Stopper
  - Storrage_System
  - Tool

hasPallet	(multiple Pallet)
index_stations	(multiple Index_Station)
inNode	(single Index_Station or Diverter or Junction)
inSensor	(multiple Sensor)
outNode	(single Index_Station or Diverter or Junction)
outSensor	(multiple Sensor)
ready	(single boolean)
rfid	(multiple RFID_Reader)
timeToPass	(single int)
turnable	(single boolean)
controlled_by	(allValuesFrom Machine_Agent, cardinality
description	(single string)
disturbance_type	(single Disturbance)
failure	(single owl:oneOf{true false})
id	(single int)
name	(single string)
setup	(single)
useChannel	(multiple LLCommunicationChannel)

Figure A.2: Part of the conveyor ontology

When the pallet leaves the conveyor, the *CA_forward_Pallet* rule will fire causing that the conveyor CC1 agent deletes the pallet from its knowledge base and informs the *outNode* agent about the approaching pallet (Figure A.3).

```
(defrule CA_forward_Pallet
    (SystemTime
        (time ?time))
    (object (is-a Machine_Agent)
        (OBJECT ?ca)
        (receive $? ?msg $?)
        (work_as ?conveyor))
   (object (is-a Conveyor)
        (OBJECT ?conveyor)
        (:NAME ?name))
    (object (is-a Message)
        (OBJECT ?msg)
        (performative "INFORM")
        (content ?pname)
        (conversation-id ?param)
        (sender ?Low_level))
    =>
    (purge ?msg)
    (bind ?outNode (slot-get ?conveyor outNode))
    (bind ?outNode_agent (slot-get ?outNode controlled_by))
    (bind $?allPallets (slot-get ?conveyor hasPallet))
    (bind ?result (run-query* is_a_Pallet ?pname))
    (while (?result next)
        (bind ?pallet (?result get pallet))
        (bind ?pallet_in_xml (to-xml ?pallet))
; for given protege pallet instance function to-xml returns string containing this pallet as xml file
        (bind ?number (member$ ?pallet  $?allPallets))
        (slot-delete$ ?conveyor hasPallet ?number ?number)
        (send_msg
; send_msg function sends a message from Protege using the proper JadeAgent
            (make-instance of Message
            (performative "SUBSCRIBE")
            (conversation-id ?conv_id)
            (sender ?ca)
            (receiver ?outNode_agent)
            (content ?pallet_in_xml)))))
```

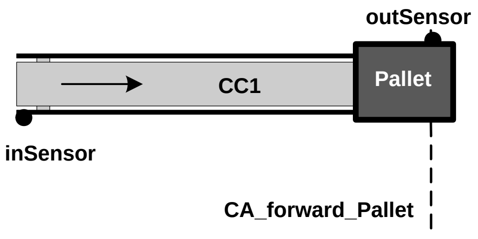


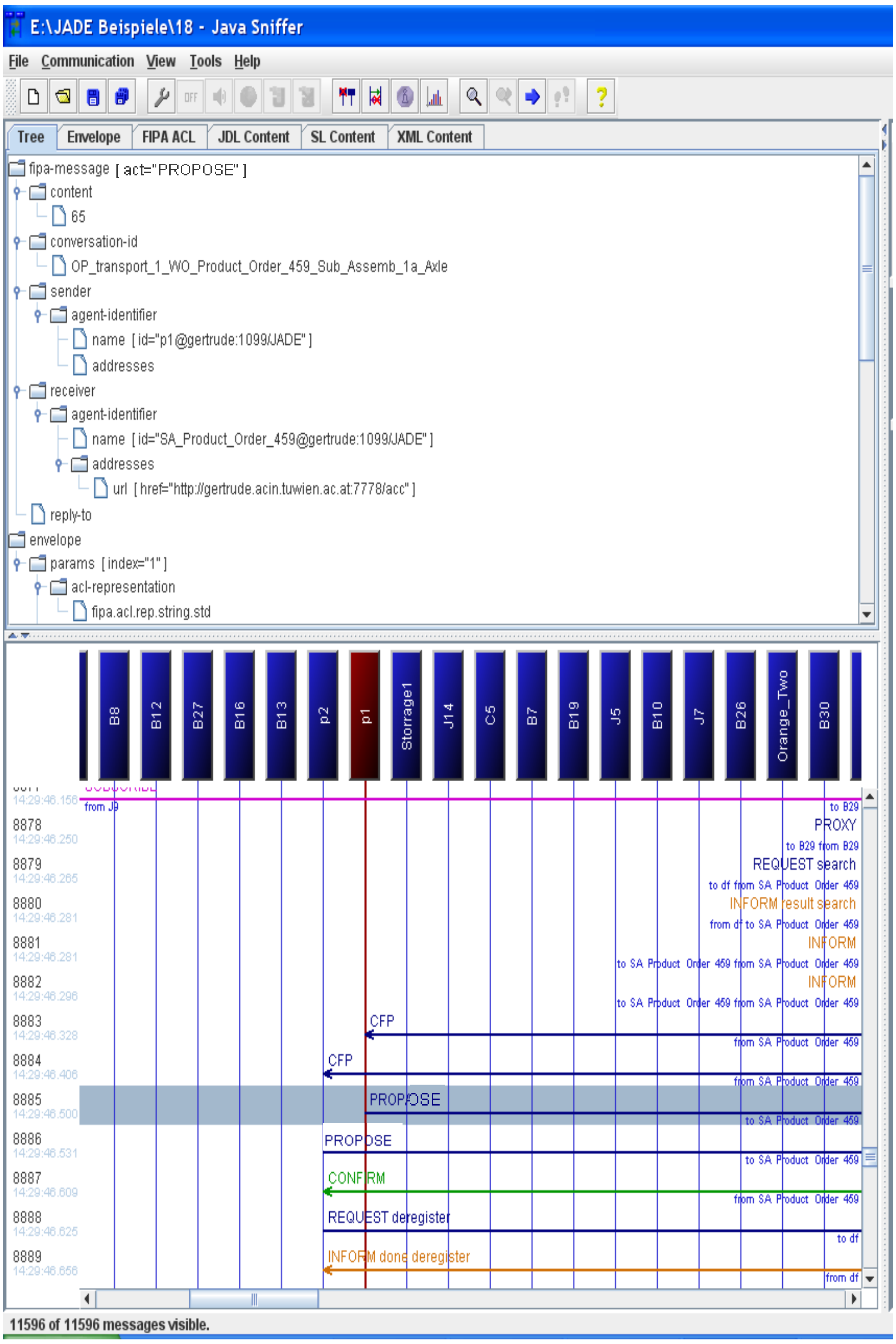
Figure A.3: Conveyor forwards a new pallet

As a second example we will present rules where the pallet agents participate in a negotiation about a transport task allocation (Figure A.4).

```
(defrule PA_negotiate_Task
    (object (is-a Machine_Agent)
        (OBJECT ?pa)
        (receive $? ?msg $?)
        (work_as ?pallet))
    (object (is-a Pallet)
        (OBJECT ?pallet)
        (busy FALSE))
    (object (is-a Message)
        (OBJECT ?msg)
        (content ?cont)
        (conversation-id ?param) ;variable contains the SA and task name
        (performative "CFP")
        (sender ?sender)
        (receiver ?pa))
    =>
    (purge ?msg)
    (bind ?operation_name (implode$(subseq$ (explode$ ?param) 2 2)))
    (bind ?conv (slot-get ?pallet located_at))
    (bind ?result (run-query* is_a_Index ?cont))
    (while (?result next)
    (bind ?java_dest (slot-get (?result get index_Station) name)))
    (bind ?si1 "")
    (bind ?re (run-query* conv))
    (while (?re next) (
        bind ?si1
            (str-cat ?si1
                (slot-get (?re get conv) name) ","
                (slot-get (slot-get (?re get conv) inNode) name) ","
                (slot-get (slot-get (?re get conv) outNode) name) ","
                (slot-get  (?re get conv) cost) ","
                (slot-get (?re get conv) turnable) "|"  )))
    (bind ?cost (call SFF.SPDetection findPath ?nextStation ?java_dest ?si1 FALSE))
    (send_msg
            (make-instance of Message
                (performative "PROPOSE")
                (sender ?pa)
                (receiver ?sender)
                (conversation-id ?operation_name)
                (content ?cost))))
```

Figure A.4: Pallet Agent negotiation rule

After the SA sends the CFP to all free pallets in the system (in our case p1 and p2 as presented in Figure A.5), the conditions for firing the rule *PA_negotiate_Task*, which is presented in Figure A.4, are reached. The related pallet agents calculate the distance of their pallets to the destination and send their results as PROPOSE messages back to the SA.



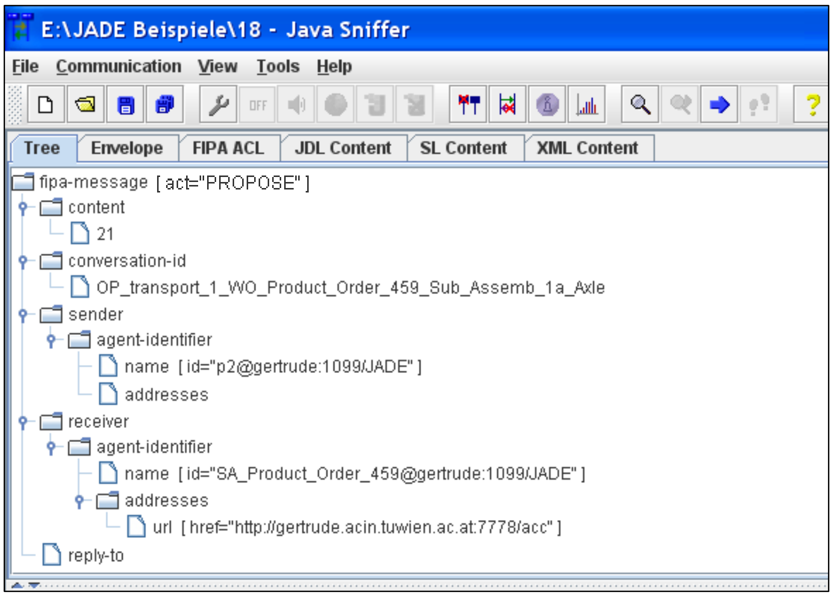
```
fipa-message [ act="PROPOSE" ]
  content
    65
  conversation-id
    OP_transport_1_WO_Product_Order_459_Sub_Assemb_1a_Axle
  sender
    agent-identifier
      name [ id="p1@gertrude:1099/JADE" ]
      addresses
  receiver
    agent-identifier
      name [ id="SA_Product_Order_459@gertrude:1099/JADE" ]
      addresses
        url [ href="http://gertrude.acin.tuwien.ac.at:7778/acc" ]
  reply-to
envelope
  params [ index="1" ]
    acl-representation
      fipa.acl.rep.string.std
```

```
14:29:46.156  SUBSCRIBE  from J9  to B29
8878  14:29:46.250  PROXY  to B29 from B29
8879  14:29:46.265  REQUEST search  to df from SA Product Order 459
8880  14:29:46.281  INFORM result search  from df to SA Product Order 459
8881  14:29:46.281  INFORM  to SA Product Order 459 from SA Product Order 459
8882  14:29:46.296  INFORM  to SA Product Order 459 from SA Product Order 459
8883  14:29:46.328  CFP  from SA Product Order 459
8884  14:29:46.406  CFP  from SA Product Order 459
8885  14:29:46.500  PROPOSE  to SA Product Order 459
8886  14:29:46.531  PROPOSE  to SA Product Order 459
8887  14:29:46.609  CONFIRM  from SA Product Order 459
8888  14:29:46.625  REQUEST deregister  to df
8889  14:29:46.656  INFORM done deregister  from df
```

11596 of 11596 messages visible.

Figure A.5. Negotiation between the SA and pallet agents

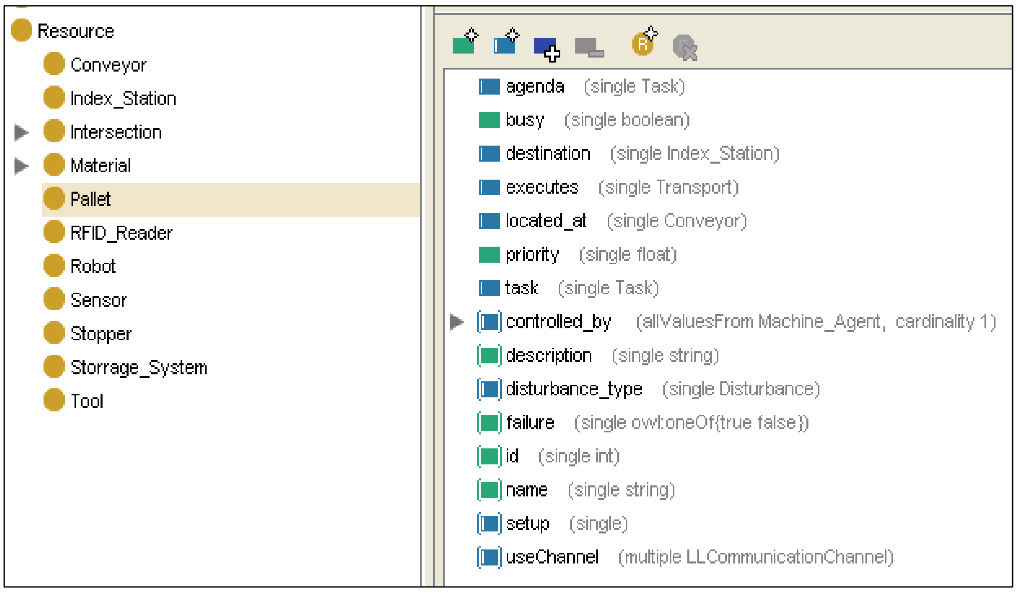
Since the distance of the p1 agent's pallet (as presented in Figure A.5) is 65 and the distance of the p2 agent's pallet (as presented in Figure A.6) is 21, the SA allocates the task >*OP_transport_1_WO_Product_Order_459_Sub_Assemb_1a_Axle*< to the p2 agent.


E:\JADE Beispiele\18 - Java Sniffer
File Communication View Tools Help
Tree Envelope FIPA ACL JDL Content SL Content XML Content

```
fipa-message [ act="PROPOSE" ]
  content
    21
  conversation-id
    OP_transport_1_WO_Product_Order_459_Sub_Assemb_1a_Axle
  sender
    agent-identifier
      name [ id="p2@gertrude:1099/JADE" ]
      addresses
  receiver
    agent-identifier
      name [ id="SA_Product_Order_459@gertrude:1099/JADE" ]
      addresses
        url [ href="http://gertrude.acin.tuwien.ac.at:7778/acc" ]
  reply-to
```

Figure A.6. The bid of the p2 agent

A part of the pallet ontology is presented in Figure A.7.

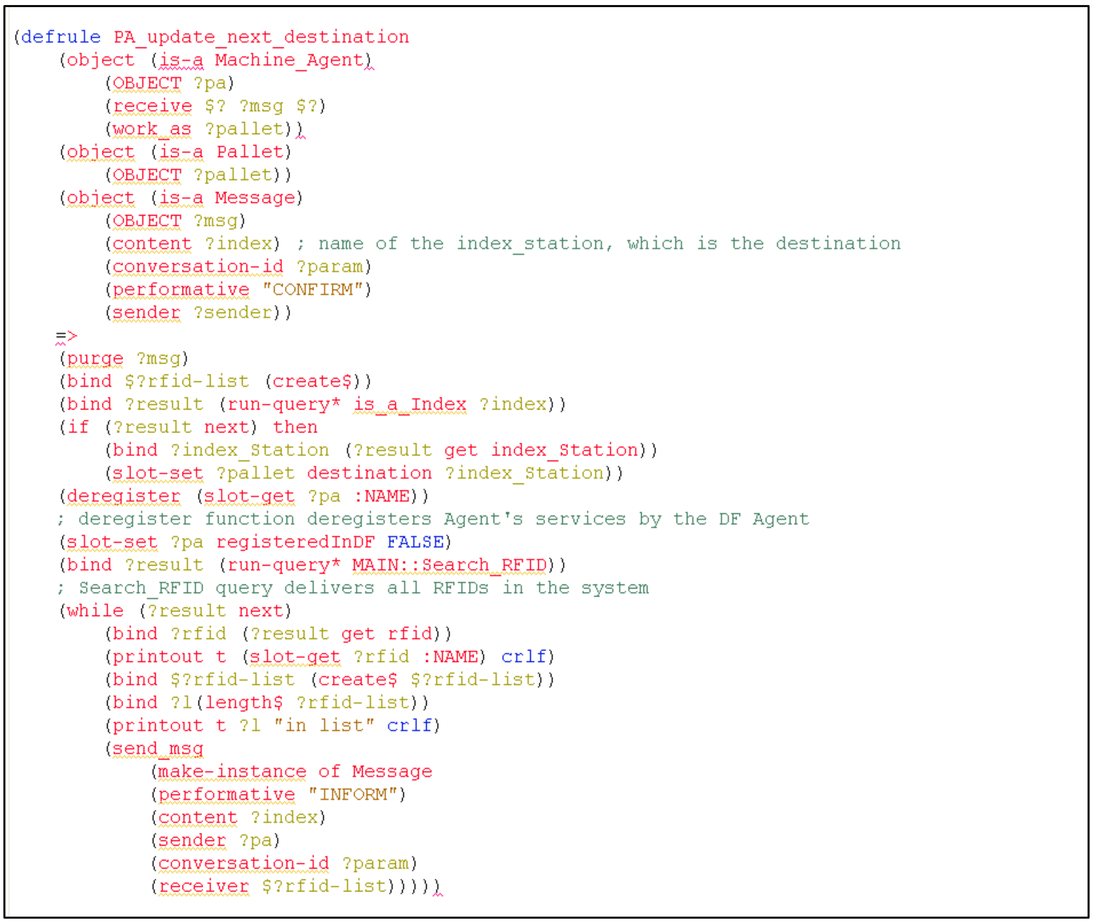


```
Resource
  Conveyor
  Index_Station
▶ Intersection
▶ Material
  Pallet
  RFID_Reader
  Robot
  Sensor
  Stopper
  Storrage_System
  Tool
```

```
agenda          (single Task)
busy            (single boolean)
destination     (single Index_Station)
executes        (single Transport)
located_at      (single Conveyor)
priority        (single float)
task            (single Task)
controlled_by   (allValuesFrom Machine_Agent, cardinality 1)
description     (single string)
disturbance_type  (single Disturbance)
failure         (single owl:oneOf{true false})
id              (single int)
name            (single string)
setup           (single)
useChannel      (multiple LLCommunicationChannel)
```

Figure A.7: Part of the pallet ontology

After the pallet agent gets the CONFIRM message about the transport task, conditions for firing the *PA_update_next_destination* rule are satisfied (Figure A.8). The pallet agent deregisters its services at the DF agent and informs all RFID agents to reprogram its pallet to the new destination, when it passes by.



```
(defrule PA_update_next_destination
    (object (is-a Machine_Agent)
        (OBJECT ?pa)
        (receive $? ?msg $?)
        (work_as ?pallet))
    (object (is-a Pallet)
        (OBJECT ?pallet))
    (object (is-a Message)
        (OBJECT ?msg)
        (content ?index) ; name of the index_station, which is the destination
        (conversation-id ?param)
        (performative "CONFIRM")
        (sender ?sender))
    =>
    (purge ?msg)
    (bind $?rfid-list (create$))
    (bind ?result (run-query* is_a_Index ?index))
    (if (?result next) then
        (bind ?index_Station (?result get index_Station))
        (slot-set ?pallet destination ?index_Station))
    (deregister (slot-get ?pa :NAME))
    ; deregister function deregisters Agent's services by the DF Agent
    (slot-set ?pa registeredInDF FALSE)
    (bind ?result (run-query* MAIN::Search_RFID))
    ; Search_RFID query delivers all RFIDs in the system
    (while (?result next)
        (bind ?rfid (?result get rfid))
        (printout t (slot-get ?rfid :NAME) crlf)
        (bind $?rfid-list (create$ $?rfid-list))
        (bind ?l(length$ ?rfid-list))
        (printout t ?l "in list" crlf)
        (send_msg
            (make-instance of Message
            (performative "INFORM")
            (content ?index)
            (sender ?pa)
            (conversation-id ?param)
            (receiver $?rfid-list)))))
```

Figure A.8: Pallet agent updates the new destination rule

B. Publications

Merdan M., Terzic I., Zoitl A., Favre-Bulle B.:"Intelligent Reconfiguration Using Knowledge Based Agent System"; 10th IEEE International Conference on Emerging Technologies and Factory Automation, Catania, Italien; 2005

Merdan M., Kordic V., Zoitl A., Lazinica A.:"Knowledge-based Multi-agent Architecture", International Conference on Computational Intelligence for Modelling Control and Automation 2006, Sydney, Australia; 2006

Merdan M., Koppensteiner G., Zoitl A., Favre-Bulle B.:"Distributed Agents Architecture Applied in Assembly Domain"; International Symposium on Knowledge and Systems Sciences (KSS), Ishikawa, Japan; 2007;

Merdan M., Koppensteiner G., Hegny I., Favre-Bulle B.:"Application of an Ontology in a Transport Domain"; IEEE International Conference on Industrial Technology, Chengdu, China, 2008

Merdan M., Moser T., Wahyudin D., Biffl S.:"Performance Evaluation of Workflow Scheduling Strategies Considering Transportation Times and Conveyor Failures"; IEEE International Conference on Industrial Engineering and Engineering Management, Singapore; 2008

Merdan M., Moser T., Wahyudin D., Biffl S., Vrba P.:"Simulation of Workflow Scheduling Strategies using the MAST Test Management System"; 10th International Conference on Control, Automation, Robotics and Vision, Hanoi, Vietnam;.2008

Vrba P., Marik V., Merdan M.:"Physical Deployment of Agent-based Industrial Control Solutions: MAST Story";IEEE SMC International Conference on Distributed Human-Machine Systems (DHMS), Athens, Greece; 2008

Merdan M., Vittori L., Koppensteiner G., Vrba P., Favre-Bulle B.:"Simulation of an ontology-based multi-agent transport system"; International Conference on Instrumentation, Control and Information Technology (SICE), Chofu, Tokyo, Japan; 2008

Merdan M., Vrba P., Koppensteiner G., Zoitl A.:"Knowledge-based Multi-Agent Architecture for Dynamic Scheduling in Manufacturing Systems"; IEEE International Conference on Industrial Informatics (INDIN), Daejeon, Korea; 2008

Hegny I., Hummer-Koppendorfer O., Zoitl A., Koppensteiner G., Merdan M.:"Integrating Software Agents and IEC 61499 Realtime Control for Reconfigurable Distributed Manufacturing Systems"; IEEE 3rd International Symposium on Industrial Embedded Systems - SIES 2008, Frankreich; 2008

Merdan M., Koppensteiner G., Zoitl A., Hegny I.:"Intelligent-Agent based Approach for Assembly Automation"; IEEE Conference on Soft Computing in Industrial Applications SMCia/08, 2008

Koppensteiner G., Merdan M., Hegny I., Weidenhausen G.:"A Change-Direction-Algorithm for distrubuted Multi-Agent Transport Systems"; IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Kagawa, Japan; 2008

Koppensteiner G., Merdan M., Zoitl A., Favre-Bulle B.:"Ontology-based Resource Allocation in Distributed Systems using Director Facilitator Agents"; IEEE International Symposium on Industrial Electronics, Cambridge, United Kingdom; 2008

Merdan M., Lepuschitz W., Hegny I., and Koppensteiner G.:„Application of a Communication Interface between Agents and the Low Level Control" 4th International Conference on Autonomous Robots and Agents. Wellington, New Zealand, 2009.