Skill-based reconfiguration of industrial mobile robots

Stefanie Angerer

School of Mathematical and Computer Sciences
Heriot-Watt University

Submitted for the degree of

Doctor of Philosophy

Heriot-Watt University

School of Mathematical and Computer Sciences

May 4, 2012

The copyright of the thesis is owned by the author. Any quotation from the thesis or use of any of the information contained in it must acknowledge this thesis as the source of the quotation or information.

Abstract

Caused by a rising mass customisation and the high variety of equipment versions, the flexibility of manufacturing systems in car productions has to be increased. In addition to a flexible handling of production load changes or hardware breakdowns that are established research areas in literature, this thesis presents a skill-based reconfiguration mechanism for industrial mobile robots to enhance functional reconfigurability.

The proposed holonic multi-agent system is able to react to functional process changes while missing functionalities are created by self-organisation. Applied to a mobile commissioning system that is provided by AUDI AG, the suggested mechanism is validated in a real-world environment including the on-line verification of the reconfigured robot functionality in a Validity Check.

The present thesis includes an original contribution in three aspects: First, a reconfiguration mechanism is presented that reacts in a self-organised way to functional process changes. The application layer of a hardware system converts a semantic description into functional requirements for a new robot skill. The result of this mechanism is the on-line integration of a new functionality into the running process.

Second, the proposed system allows maintaining the productivity of the running process and flexibly changing the robot hardware through provision of a hardware-abstraction layer. An encapsulated Reconfiguration Holon dynamically includes the actual configuration each time a reconfiguration is started. This allows reacting to changed environment settings. As the resulting agent that contains the new functionality, is identical in shape and behaviour to the existing skills, its integration into the running process is conducted without a considerable loss of productivity.

Third, the suggested mechanism is composed of a novel agent design that allows implementing self-organisation during the encapsulated reconfiguration and dependability for standard process executions. The selective assignment of behaviour-based and cognitive agents is the basis for the flexibility and effectiveness of the proposed reconfiguration mechanism.

Für meine Eltern Irma und Konrad Angerer.

Acknowledgement

In the course of investigating the reconfiguration of mobile robots I have enjoyed the help and encouragement of a number of great people whom I would like to thank.

First and foremost, I would like to thank my supervisors Professor Dr. Rob Pooley and Professor Ruth Aylett. Their constant support and the valuable discussions contributed decisively to this thesis. Thank you for all the guidance and encouragement over the last years. Further, I like to thank the external examiner Dr. Subramanian Ramamoorthy and the internal examiner Dr. Patricia Vargas for their most valuable input.

The research presented in this thesis has been funded by the Technology Development Production of AUDI AG in Ingolstadt. I would like to thank the department for the funding and technical support of this work and the great possibility to setup a mobile commissioning system that is able to reconfigure itself. Dr. Jens Bunte deserves my special thank for his support as his advices were not only invaluable but as well taken. Thank you for teaching me what professional life is all about. Further, I would like to thank the head of the department Dr. Klaus Koglin for his confidence in this work.

A main part of this research was carried out at Ingolstadt Institute for Applied Research under the guidance of Professor Dr. Johann Schweiger. Besides his steady encouragement, he taught me for life about enthusiasm, resilience, and integrity.

I would like to express my thanks to my colleagues and students who shared their thoughts on my thesis, most notable Dr. Andreas Hermann, Dr. Stephan Matzka, and Dr. Zsolt Husz.

I am obliged to my partner Michael for his patience and his unbreakable faith in me. I would like to extend a warm thank to him, my parents, and my sister who are the place in life to share joy and sorrows.

Stefanie Angerer

May 4, 2012

Table of Contents

List of Tables

List of Figures

List of Equations

Listings

Glossary

ACL Agent Communication Language, 64

ADACOR Adaptive Holonic Control Architecture, 41

AGV Automated Guided Vehicle, 34

AMS Agent Management System, 64

AOSE Agent-oriented Software Engineering, 61

API Application Programming Interface, 79

BDI Belief Desire Intention, 12

C-SA Composite Skill Agent, 80

CA Coordination Agent, 33

CAD Computer Aided Design, 57

CFP Call For Proposal, 69

CNC Computer Numerically Controlled, 30

CNP Contract Net Protocol, 42

CoBASA Coalition Based Approach for Shop Floor Agility, 33

DF Directory Facilitator, 12

EAS Evolvable Assembly Systems, 48

ERP Enterprise Resource Planning, 35

FIPA Foundation of Intelligent Physical Agents, 64

FSM Finite State Machine, 114

GSA Generic Skill Agent, 12

GTA Generic Task Agent, 12

HMS Holonic Manufacturing System, 4

I-EA Execution Agent, 12

I-IA Initiator Agent, 12

IEEE Institute of Electrical and Electronics Engineers, 68

ISO International Organisation for Standardisation, 163

JADE Java Agent Development Environment, 63

LOCOBOT Low cost robot co-workers, 6

LPS Local Perceptual Space, 18

LTL Linear Temporal Logic, 22

MARA Multi Agent Resource Allocation, 69

MAS Multi Agent System, 31

MAS4AMR Multi Agent System for Auto Mobile Robot, 18

MASCADA Manufacturing Control Systems Capable of Managing Production Change and Disturbances, 38

MES Manufacturing Execution System, 35

MobComm Mobile Commissioning, 1

MRA Manufacturing Resource Agent, 33

NSD New Skill Description, 12

NSID New Skill Input Data, 12

OC Organic Computing, 51

ODP Organic Design Pattern, 52

OH Operational Holon, 41

P'n'P Plug and Produce, 58

PA Process Agent, 80

PABADIS Plant Automation Based on Distributed System, 35

PH Product Holon, 41

PLC Programmable Logic Controllers, 32

PROMISE Product Oriented Manufacturing Systems for Reconfigurable Enterprises, 35

PROSA Product, Resource, Order and Staff Architecture, 38

RA Resource Agent, 80

RAP Reactive Action Package, 17

RH Reconfiguration Holon, 12

RIA Restore Invariant Approach, 51

SA Skill Agent, 80

SH Standard Holon, 12

SHAGE Self-Healing, Adaptive, and Growing Software, 21

SIARAS Skill-Based Inspection and Assembly of Reconfigurable Automation Systems, 56

SME Small and Medium Enterprises, 58

SO-EAS Self-Organising Evolvable Assembly System, 49

SPA Sense Plan Act, 17

SuOC System under Observation/Control, 51

TA Task Agent, 80

TH Task Holon, 41

VC-PA Process Agent for Validity Check, 131

XML eXtensible Markup Language, 66

YAMS Yet Another Manufacturing System, 68

List of Publications

Parts of this thesis are based on the following conference papers:

- S. Angerer, R. Pooley. Dependable Reconfiguration of Mobile Manufacturing Systems. Proceedings of the 14th IASTED International Conference on Robotics and Application, Cambridge, USA, 2009.

- S. Angerer, R. Pooley, R. Aylett. MobComm: Using BDI-Agents for the Reconfiguration of Mobile Commissioning Robots. Proceedings of the 6th IEEE International Conference on Automation Science and Engineering (IEEE CASE), Toronto, Canada, 2010.

- S. Angerer, R. Pooley, R. Aylett. Self-Reconfiguration of Industrial Mobile Robots. Proceedings of the 4th IEEE International Conference on Self-Adaptive and Self-Organising Systems (IEEE SASO), Budapest, Hungary, 2010.

- S. Angerer, C. Strassmair, M. Röttenbacher, M. Stähr, N. Robertson. Give me a hand - the potential of mobile assistive robots in automotive logistics and assembly applications. Proceedings of the 4th Annual IEEE International Conference on Technologies for Practical Robot Applications (IEEE TePRA), Boston, USA, 2012.

Chapter 1

Introduction to Reconfigurable Industrial Mobile Robots

In this thesis a skill-based reconfiguration mechanism for industrial mobile robots is proposed. The presented MobComm (Mobile Commissioning) approach utilises the holonic paradigm and has a basic division between a Standard Holon for routine executions, and a Reconfiguration Holon, where reconfiguration tasks are accomplished to provide high productivity. The goal of every reconfiguration is to generate a new Composite Skill Agent, containing the answer to functional process changes. The robot hardware is able to immediately use this new skill and thus to satisfy the changed manufacturing process needs.

The motivation of the thesis is explained in section 1.1, followed by the vision of industrial mobile robots in automotive industry in section 1.2. The research structure of the thesis is presented in section 1.3. A short overview of the scientific fields of influence is given in section 1.4, followed by the thesis overview and organisation in section 1.5.

1.1 Motivation

Car manufacturers, like most manufacturing companies, face a rising mass customisation of their products. Mass customisation requires the manufacturing system to be highly flexible [Pollard et al., 2008] because of the high trim level of cars, shortened product life cycles, and an instant satisfaction of customers' demands [Bussmann and Schild, 2000]. As presented in figure 1.1, the car industry itself changed tremendously in the last decades.

Besides increased mass customisation, the decreased time-to-market, and the increased level of complexity are the most considerable changes in this sector.

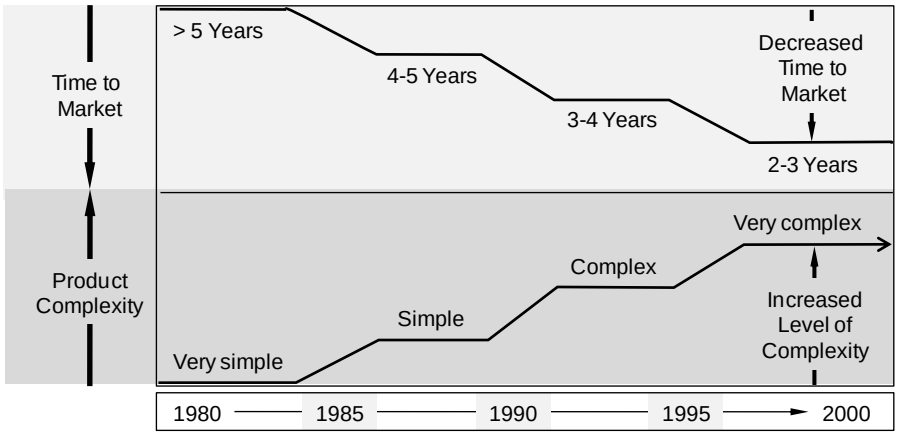


Figure 1.1: The evolution of car industry in the last decades. Adapted from: [Bi et al., 2008].

Following [Bi et al., 2008], the time-to-market decreased from over 5 years in the 1980s to around 2 years in year 2000 whereas the complexity of the cars advanced from very simple to very complex. These characteristics of automotive industry require changes in a broad set of operational sequences in the factories. This thesis contributes to a more flexible manufacturing component level.

Mass customisation, decreased time-to-market, and the increased level of complexity require changes in hardware systems, control structures and software engineering processes for manufacturing systems. As stated in [Lepuschitz et al., 2010], current and future manufacturing systems must be able to rapidly reconfigure under changed environment conditions.

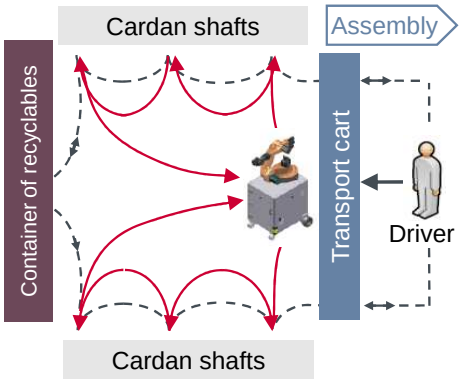
Not only must a superior manufacturing system provide a high degree of flexibility and reconfigurability, but the single manufacturing components are required to be more adaptable to changes in the production process as well. Single manufacturing systems are traditional industrial robots, conveyors, or drilling machines, to name just a few examples.

Even if the research area of mobile robots provides already a wide range of reconfiguration tools for a changed environment like a dynamic behaviour adaptation of mobile robots (cf. Saphira [Konolige and Myers, 1996]), mechanisms given in literature cannot be applied to industrial mobile robots as they face a different set of requirements for their productive use in factories.

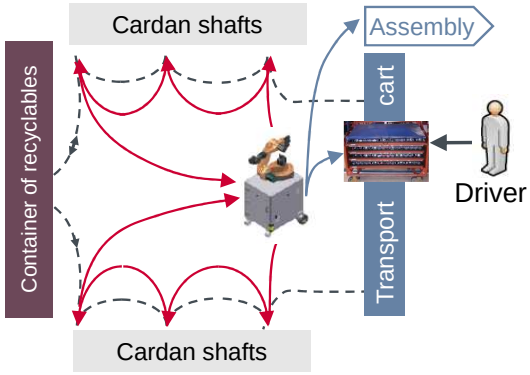
As service robots mostly aim to dynamically adapt to unknown or upcoming envi-

ronment settings, temporal constraints or the need of a high predictability do not feature significantly in this area. In contrast to that, industrial mobile robots face an environment with a strict cycle time and exact process descriptions. For this reason, industrial mobile robots must provide a very high level of robustness and predictability during standard process execution.

But nevertheless the use of industrial mobile robots is very reasonable to automate for example logistic pick and place tasks, called commissioning. An example application for a mobile robot in logistics is given in figure 1.2(a) where the commissioning of cardan shafts is presented. The robot robustly provides the functionality to pick and place different types of cardan shafts. It ultimately places these components in the provided transport cart in the order as desired by the assembly process.



(a) Example process with industrial mobile robot: Commissioning cardan shafts.



(b) Requirement of process change: Follow transport cart.

Figure 1.2: Use case for mobile robots in car manufacturing: Example commissioning of cardan shafts.

Additionally to the compliance of industrial requirements like robustness or availability, the mobile robot must be able to dynamically react to process changes that occur in the context of model changes or further derivatisation of existing models. An example process change is described in figure 1.2(b) with the required tracking of a transport cart to the assembly line.

Due to the dynamic environment, this robust mobile robot must be adaptable to the new process requirement by a skilled worker to avoid follow up costs for software changes. Figure 1.3 overviews the total commissioning process including manual and automated workplaces. A reconfiguration of the mobile robot is required if the set of provided functionalities (i.e. Skill Agents) is not sufficient any more to comply with the

defined manufacturing process. This type of functional reconfigurability cannot be covered in the context of the set industrial requirements by approaches given in literature.

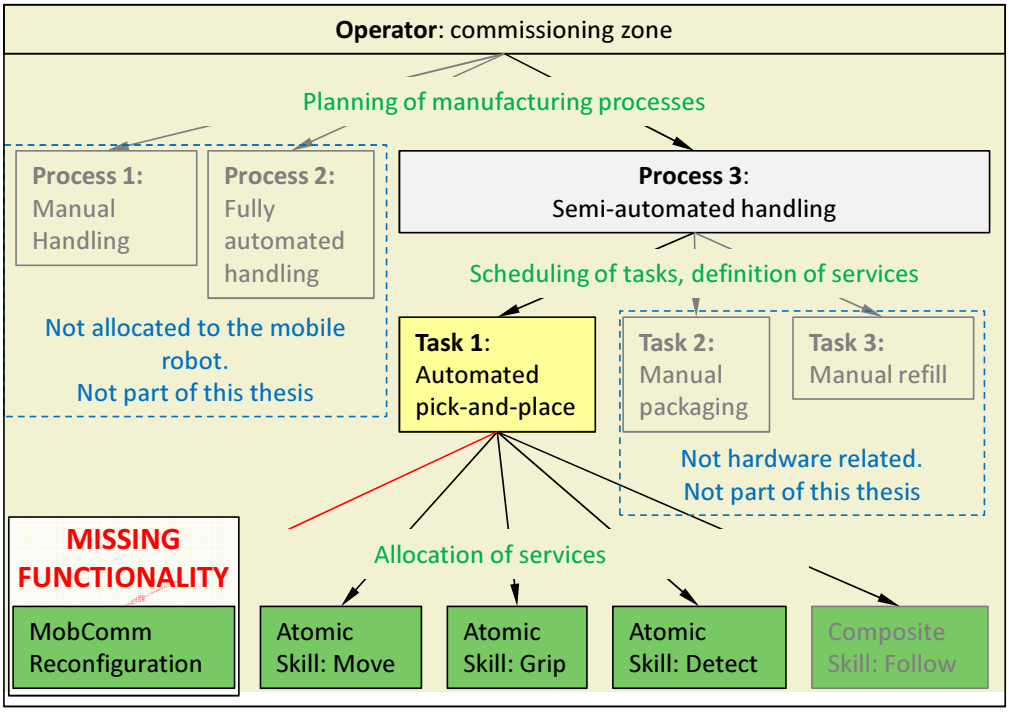


Figure 1.3: Integration of MobComm reconfiguration in total commissioning process.

The need of a robust standard process execution and a dynamic skill reconfiguration without programming effort after functional process changes motivates the proposed skill-based reconfiguration mechanism. The presented work contributes to a higher benefit of flexible mobile robots in car factories of the future.

Following the 2010 Technology Market Survey of Gartner, mobile robots are an emerging technology able to be adopted by the mainstream in more then ten years [Chip Online, 2010]. As highlighted in figure 1.4, mobile robots have already passed the technology trigger and are approaching the peak of inflated expectations.

Flexibility applicable to industrial mobile robots is investigated in different research areas. Reconfigurable, flexible, holonic, and evolvable manufacturing systems are discussed in section 2.2. Especially the use of Holonic Manufacturing Systems (HMS) leads to a high level of flexibility in production flow control. By the application of a hybrid manufacturing control with both hierarchical and heterarchical structures, flexible resource management and the dynamic allocation of production units can be provided by this research area (e.g. [Van Brussel et al., 1998]). A flexible adaptation of temporal process changes can be

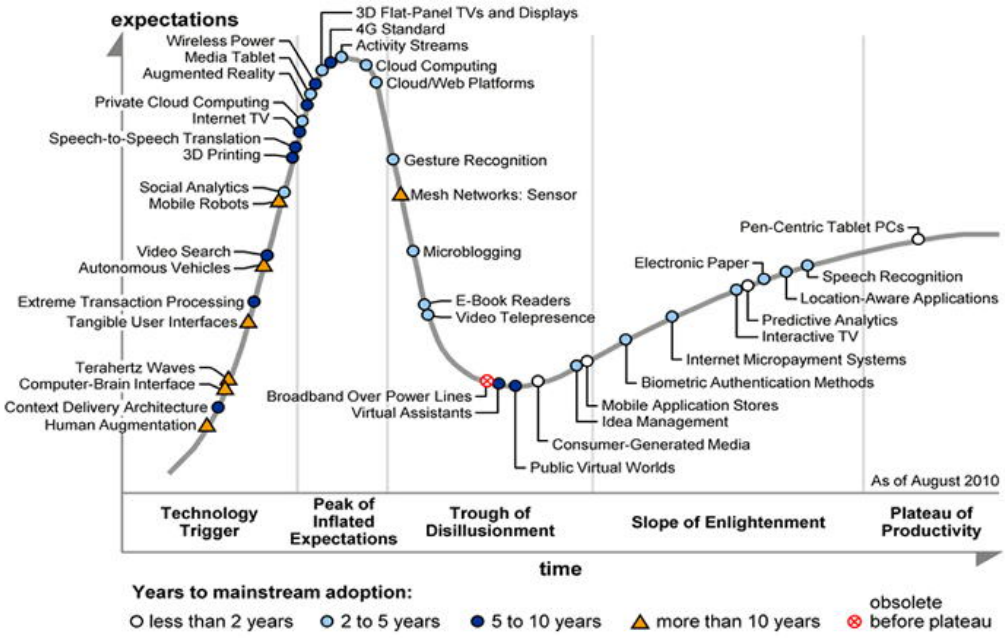


Figure 1.4: Market survey of Gartner Inc. including the technology level of mobile robot technology. Source: [Chip Online, 2010].

further reached through dynamic scheduling mechanisms within the holonic principle.

A dynamic reaction to hardware failures in manufacturing systems is pictured in a set of approaches such as the Restore Invariant Approach [Guedemann et al., 2006]. In this approach, a hardware failure violates a logical formula, the invariant, and allows to dynamically restore it by the allocation of a different task to this formula.

But neither HMS nor the Restore Invariant Approach are able to react to functional process changes on manufacturing component level as desired for industrial mobile robots in car manufacturing. On that account, this dissertation presents a reconfiguration mechanism for industrial mobile robots using a novel approach to react to these changes. The motivation for the MobComm approach and the different types of flexibility are summarised in figure 1.5.

FMS mostly reach their goals by the use of agent technology that is characterised by autonomy, pro-activity, and location-independence [Huhns and Buell, 2002, Wooldridge, 1998]. In turn, applying agent technology leads to a rise of system and program complexity, and proofs of reliability are harder to provide in real-world applications. This is one reason for the lack of real automation implementations of agent technology as described in [Leitão and Restivo, 2008]. This dissertation includes the implementation of the proposed MobComm reconfiguration mechanism. The real-world evaluation as described in 7 is

accomplished at the German car manufacturer Audi where test environment and hardware are provided.

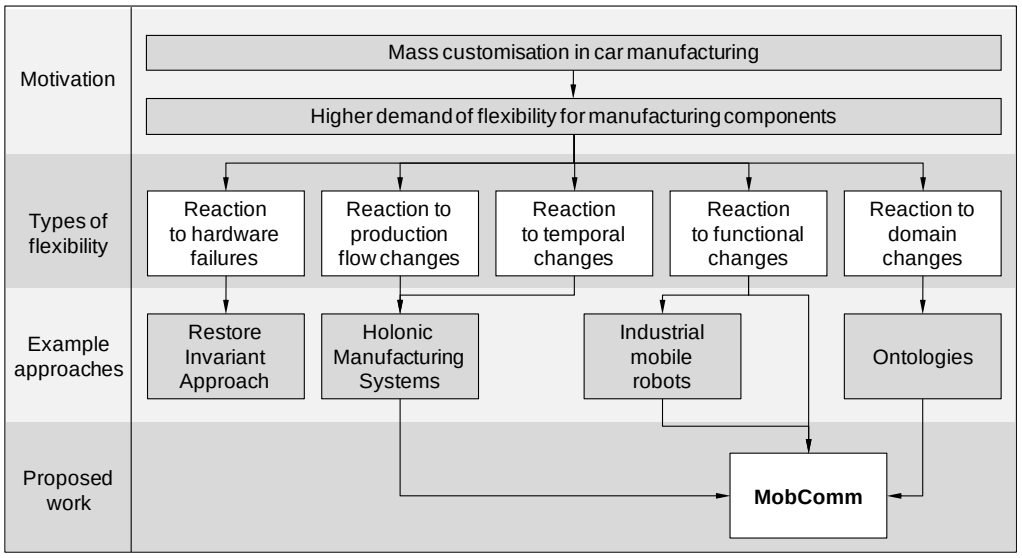


Figure 1.5: Motivation for MobComm reconfiguration mechanism.

1.2 Vision

The vision behind this thesis agrees with the market survey presented in figure 1.4 and imagines a mainstream adaptation of mobile robots. The distribution of industrial mobile robots is viewed as the enhancement of traditional industrial robots in car manufacturing.

Based on a holonic or flexible manufacturing control, an armada of mobile robots is available for the use in different areas in car manufacturing. These mobile robots are applicable to a set of different applications in the factory. As presented in figure 1.6, the superior manufacturing system manages the necessities of mobile robots in the different areas and distributes tasks to the individual robots.

The range of tasks executed by an industrial mobile robot, consisting of a mobile platform, a manipulator, a gripper and a sensory system, is broad and not limited to the given examples. Logistic handling or pick-and-place tasks are the core applications, followed by worker assistance or bring-and-delivery tasks between different areas in the factory.

Especially tasks of worker assistance for industrial mobile robots are focused on in the LOCOBOT (Low cost robot co-workers) project funded by the European Commission's 7th Framework Programme [Profactor GmbH, 2010]. With partners like the Heriot-Watt

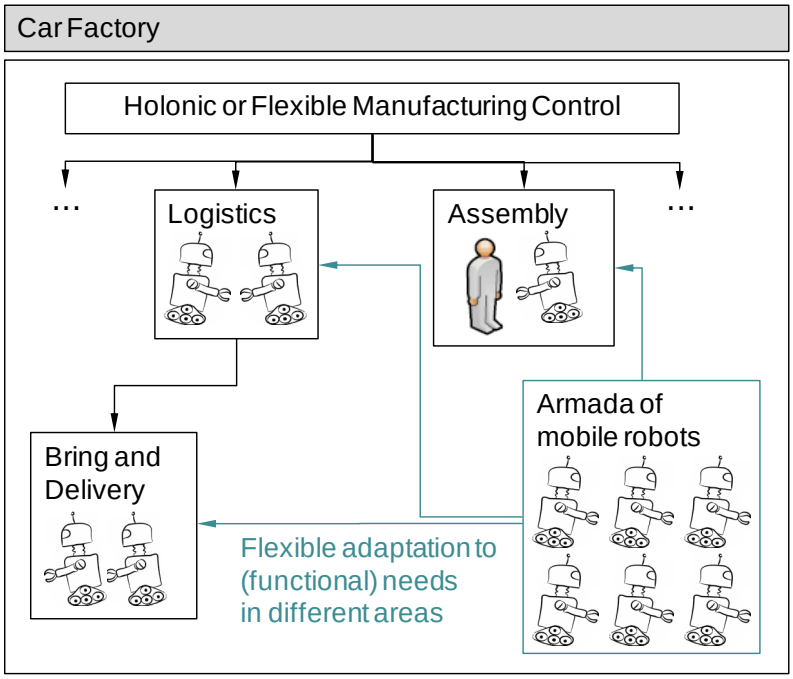


Figure 1.6: Vision for industrial mobile robots in car manufacturing.

University, the University of Edinburgh and Audi, a modular and collaborating mobile robot is investigated aiming at combining reconfigurability in manufacturing with cost effectiveness.

To flexibly adapt the functionalities needed by the mobile robots, a scheduling and reconfiguration mechanism is required. Generic mobile robots have to be able to self-adapt to the upcoming tasks in the factory. A step towards this self-adaptation of generic industrial mobile robots in a productive environment is the reconfiguration mechanism proposed in this thesis.

1.3 Research Structure

The research overview is structured into thesis contribution and hypothesis, resulting research objectives, and finally a set of research assumptions, as presented in the following sections.

1.3.1 Contribution and Hypothesis

The contribution of this thesis can be divided into three aspects: the proposal of a novel reconfiguration mechanism after functional process changes, a hardware-abstract system

with maintenance of productivity, and the novel agent design for dependability and self-organisation in productive environments.

Reconfiguration mechanism after functional process changes

The presented reconfiguration mechanism is based on [Angerer et al., 2010b] and is a novelty regarding its self-organised reaction to functional process changes. Extending the hardware-related or temporal reconfiguration aspects as given in [Guedemann et al., 2006] or [Frei et al., 2007c], MobComm proposes the conversion of a semantic description of a functionality into a new agent that represents this robot skill. This reconfiguration mechanism, implemented on manufacturing component level, contributes to the extension of manufacturing flexibility.

Hardware-abstracted system with maintenance of productivity during reconfiguration

The proposed system is novel due to its basic structure of two separated parts in the system, implemented as Reconfiguration and Standard Holons allowing the complete separation of task execution and reconfiguration for the maintenance of productivity. The presented system extends the suggestion given in [Angerer and Pooley, 2009], and is based on [Angerer et al., 2010a].

The segregation and reintegration of reconfiguration results is based on the holonic principle as given in ADACOR [Leitão and Restivo, 2008] or PABADIS [Feng et al., 2007]. Adapted from the hybrid control structures of Holonic Manufacturing Systems, where both robust hierarchical and adaptive heterarchical structures can be applied, the MobComm mechanism consists of a robust Standard Holon and creates a changed skill configuration in a heterarchical organised Reconfiguration Holon.

Besides maintaining productivity, the novel aspect is the configuration-independent reconfiguration while providing a hardware abstraction layer. Configuration independence is achieved by the on-line access of reconfiguration knowledge and the migration of agent clones, whereas hardware abstraction is based on the use of a resource agent layer providing defined interfaces for a broad range of hardware components.

Agent design for dependability and self-organisation in productive environments

According to the requirements of a productive environment, a novel agent design for dependability and self-organisation is presented. Corresponding the definition 2.12 on page 45, self-organisation includes self-management structure adaptation and the provision of decentralised control. The Standard Holon, used for routine executions, is realised by behaviour-based agents using a service-based communication. This aspect of the MobComm agent ensures dependability for the executed processes in cycle time. Agents used in Reconfiguration Holon, however, are designed as BDI-agents and able to support self-organisation and self-awareness. The reconfiguration mechanism draws on reasoning and planning of the BDI principle, nevertheless the outcome of a successful reconfiguration is a behaviour-based agent, suitable for the use in Standard Holon. By using this novel combination of behaviour-based and BDI aspects, the agent design in MobComm is suitable for a dependable execution of tasks and a self-organised handling of process changes in productive environments.

Based on these three aspects of thesis contribution, the research hypothesis is formulated as follows:

1. Functional process changes are inserted as semantic descriptions by a user. The proposed reconfiguration mechanism transforms these descriptions self-organised into the desired robot functionality.

2. MobComm reconfiguration mechanism can be executed without disturbing the running manufacturing process.

3. The reconfigured robot functionalities provide a high dependability for their permanent use in the standard process.

4. The used mobile robot is expandable by further hardware components that implement the MobComm specification.

1.3.2 Research Objectives

This section presents the research questions derived from the research hypothesis. By using the provided contribution and hypothesis, the main thesis question is determined as follows:

How is it possible to reconfigure industrial mobile robots self-organised in their hardware limits after functional process changes?

The thesis question includes that the reaction to functional process is handled self-organised by the reconfiguration mechanism. This covers a self-managed execution of computational steps without the need of external control by following the definition of self-organisation on page 45. The hardware components of the provided mobile robot are further regarded as fixed and thus not used as a source of reconfiguration or failure.
Based on the thesis question and the research hypothesis, a set of research objectives is outlined as goals for this work:

Objective 1: Self-organised reconfiguration with maintenance of productivity during reconfiguration.

Objective 2: Dependable integration of new skills.

Objective 3: Handling of functional process changes with an abstraction of given hardware.

1.3.3 Research Assumptions

MobComm requires a set of system requirements and environment premises to be implemented as stated in the thesis contribution.

Assumption 1: The used hardware operates as specified. No communication or bus errors are regarded during reconfiguration.

Assumption 2: No hardware failures occur during reconfiguration. Hardware breakdowns are not regarded in this approach.

Assumption 3: Descriptions of new functionalities are inserted by the operator and regarded as semantically correct. No feedback loop between the user and the resulting system configuration after a reconfiguration is provided. This feedback loop is not within the scope of the thesis.

Assumption 4: New functionalities are within the example domain that allows to keep system ontology valid during reconfiguration. In case a new domain is desired, the according ontology must to be updated by using expert knowledge.

The list of research assumptions finalises the research structure section that has focused on the thesis contribution and the deduction of a set of research objectives. While an overview of structure and organisation of this work is given in section 1.5, the following section introduces the scientific fields of influence.

1.4 Scientific Fields of Influence

To detail the relationship between the research objectives and the approaches provided in literature, a set of research fields is evaluated in chapter 2. Figure 1.7 introduces the relation between research objectives and related work. The key fields for MobComm are mobile robots (cf. section 2.1) and manufacturing systems (cf. section 2.2) including their self-organisation (cf. section 2.2.2). As it is desired that a mobile robot is able to handle process changes, mobile robot research is of high interest. Further the used robot hardware, environment and executed processes are highly related to manufacturing systems. Comprehensively, related work concerning agent-oriented software engineering techniques is surveyed in section 2.3, as the relevant approaches presented in section 2.1 and 2.2 emphasise the use of agent technology for MobComm.

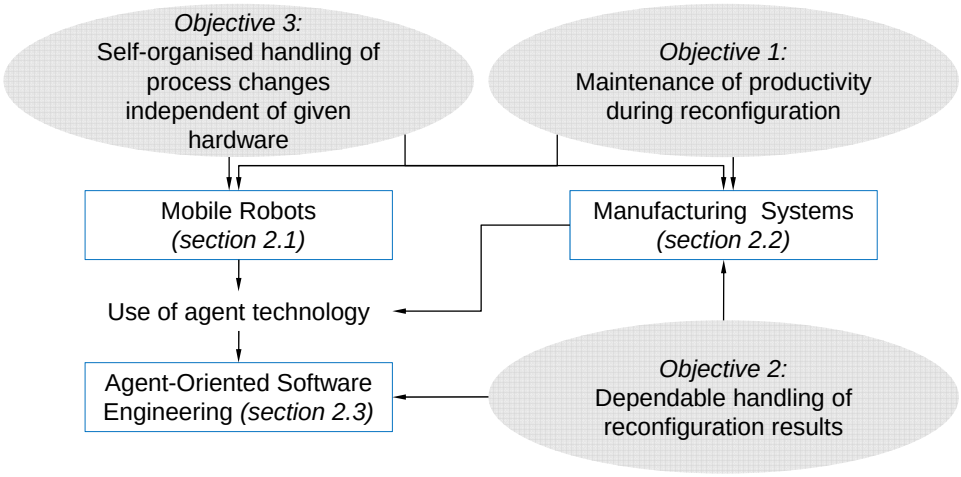


Figure 1.7: Relationship between research objectives and related work.

1.5 Thesis Overview and Organisation

In this thesis a reconfiguration mechanism dealing with functional process changes is proposed for industrial mobile robots. The scope of the dissertation comprises the design of

a reconfiguration mechanism after functional process changes, followed by its implementation and evaluation.

A complete overview of the MobComm approach is presented in figure 1.8. The system is divided into a single Standard Holon (SH) and [1..n] Reconfiguration Holons (RH). Holons are autonomous and at the same time co-operative building blocks in hierarchies as futher defined in definition 2.11 on page 37.

The Standard Holon maps the routine executions of the system in cycle time and is divided into four layers. Process and Task Layer provide a global and local schedule for the actual process. Skill Agents form the key layer as they hold all functionalities executable in the system, encapsulated in agents. Resource Layer, meanwhile, represents the connection to the real-world and includes the interfaces to the underlying robot hardware. The semantic level of the Standard Holon is represented by an ontology that contains all domain vocabularies.

The insertion of a New Skill Description (NSD), arising from a process change in the manufacturing system, leads to the initialisation of a Reconfiguration Holon. Compared to the hierarchical structure of Standard Holon, agents in Reconfiguration Holon are organised heterarchical and follow the Belief-Desire-Intention (BDI) principle. As pictured in figure 1.8, the main purpose of a reconfiguration is the processing of the NSD in Reconfiguration Holon and its reintegration as a new Composite Skill Agent in Standard Holon. The according flow of reconfiguration is summarised in figure 1.9.

The insertion of a NSD is handled and analysed by a Generic Task Agent (GTA), described in chapter 4, and is sent forward to the Reconfiguration Holon for reconfiguration purposes. Inside a Reconfiguration Holon, an Initiator Agent (I-IA) defines its goals and beliefs according to the incoming NSD. All Execution Agents (I-EA) link themselves to the knowledge and agent behaviour of Standard Holon to execute the reconfiguration mechanism. The outcome of the algorithm, executed by collaborating I-EAs, is a New Skill Input Data (NSID). The NSID is comparable to a construction plan of the new agent which includes all knowledge about the new Composite Skill Agent. This structure cannot be used until a Generic Skill Agent (GSA) converts this data into the Composite Skill Agent format. The resulting agent is identical in shape and behaviour to the already existing ones in Standard Holon. Thus, it can be easily integrated into Standard Holon. For an immediate use, its service has only to be registered at the Directory Facilitator (DF), the yellow pages of Standard Holon. Every reconfiguration process finishes with a

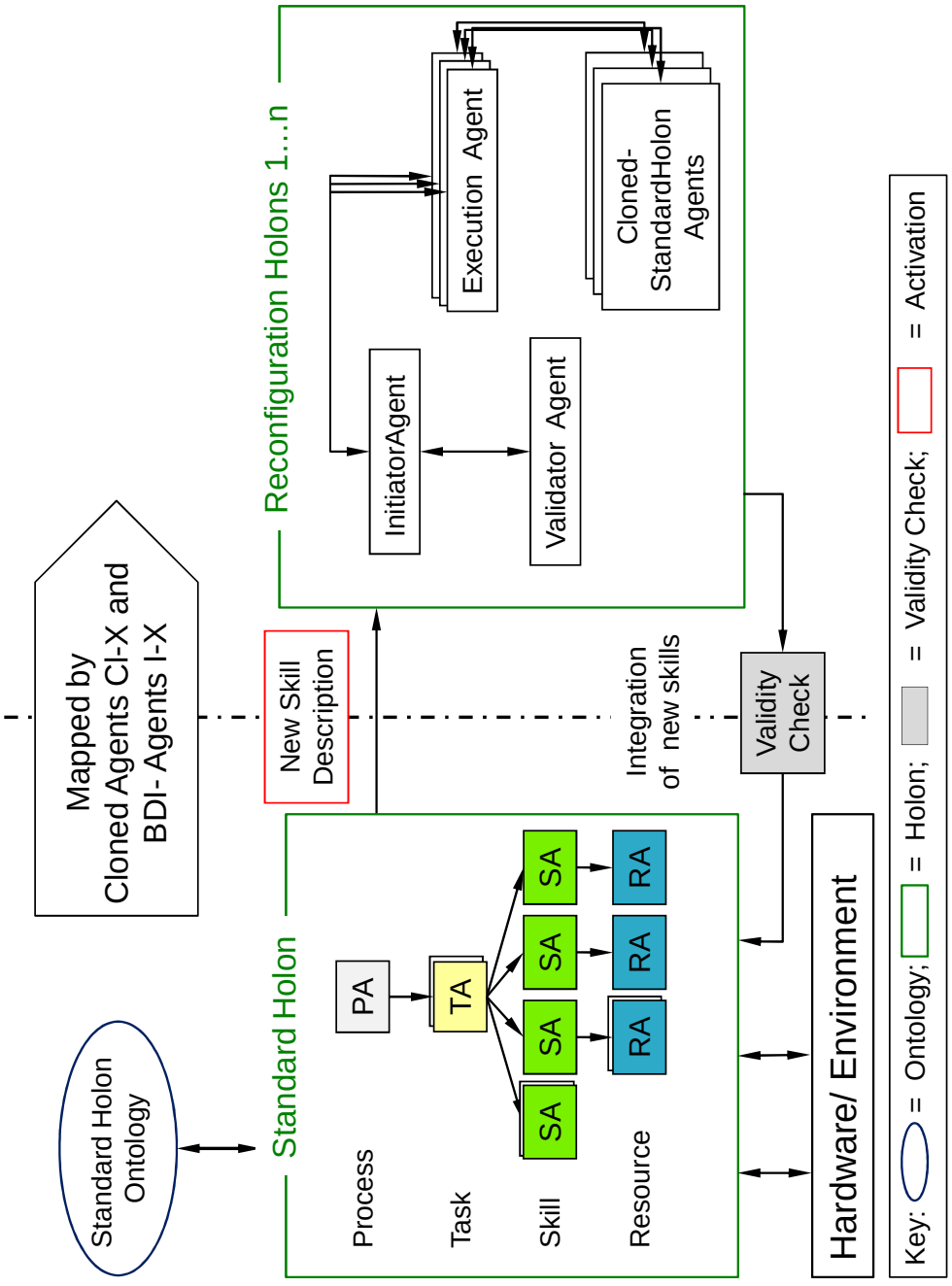


Figure 1.8: Block diagram of MobComm reconfiguration approach.

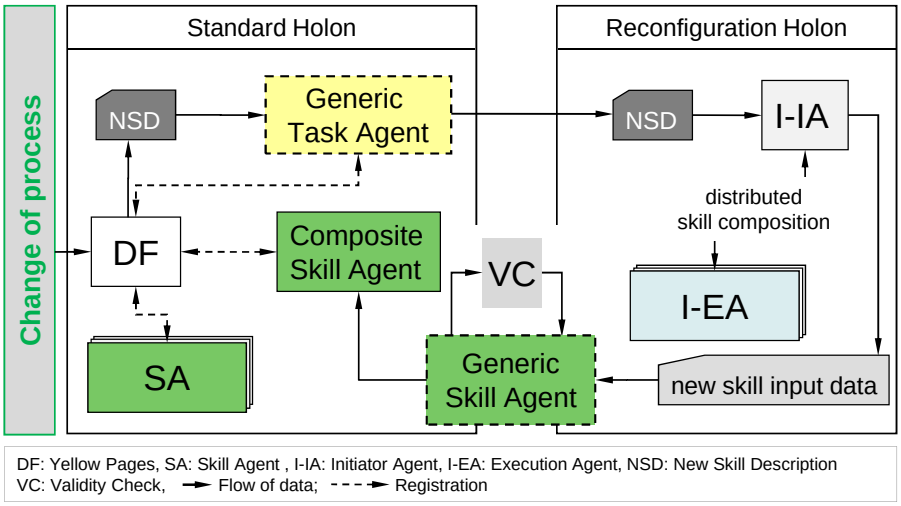


Figure 1.9: Overview of reconfiguration flow and the basic principles of the MobComm reconfiguration.

Validity Check that ensures that no unwanted or harmful operations are executed by the mobile robot system.

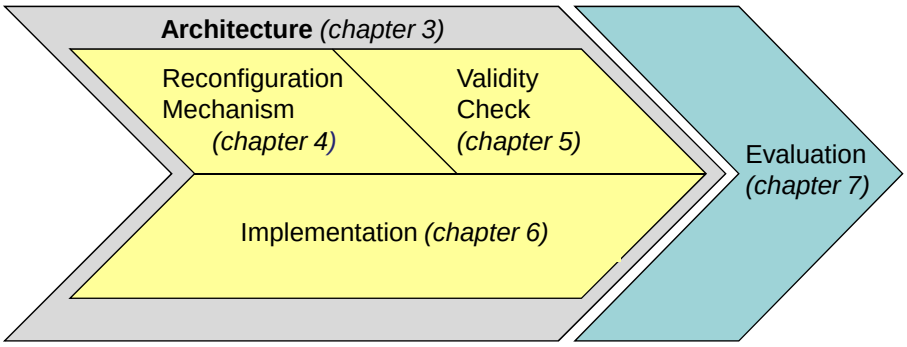


Figure 1.10: The organisation of the dissertation.

Passing over from the flow of reconfiguration to the organisation of the described system in this thesis, chapter 2 reviews related work as described in section 1.4. The main part of the thesis is organised in chapters corresponding to figure 1.10 and basically divided into three parts.

The MobComm reconfiguration as the core part starts in chapter 3 with the supporting architecture and focuses on the reconfiguration mechanism itself in chapter 4 and the Validity Check in chapter 5. The resulting MobComm implementation is given in chapter 6 and the MobComm approach is completed by its evaluation in chapter 7.

Chapter 2

Literature Review

After having presented the thesis contribution and the corresponding research objectives, the related areas of research, as introduced in figure 1.7, are discussed and evaluated in this chapter.

The research in mobile robotics is reviewed in section 2.1 as closely related to the hardware of the commissioning robot used. The area of manufacturing systems, the focus of section 2.2, is important as many issues solved in this area, like reconfigurability or flexibility, are related to the given research objectives. Surveying the agent-oriented software engineering in section 2.3, indications about the implementation aspects in this work are given.

All approaches are evaluated concerning their relevance to this work, but also concerning the limitations which need to be overcome in order to provide the answers to the research question. The conclusion of the literature review, however, results in a set of research and supportive tasks that arise from the discussion of the single fields of interest and allow the evaluation of the reconfiguration mechanism by the compliance of these tasks.

2.1 Mobile Robot Systems

Even though mobile robots and their control systems already have a long history, they still raise many unsolved questions following the survey of future challenges in robotics [Bekey et al., 2008]. As given in the motivation in section 1.1 and more detailed in figure 1.2, the industrial mobile robot used in this thesis must be able to react to functional process

changes while providing a robust execution of standard processes. The hardware of the industrial mobile robot in this thesis, a mobile commissioning system, belongs to the autonomous mobile robots, as defined in the following:

Definition 2.1 (Autonomous Mobile Robot) *An autonomous mobile robot is a machine able to do environmental navigation on its own without a human directly manipulating. The robot must be able to perceive its surroundings through different kinds of sensors and initiate appropriate actions in that environment through actuators to achieve its designed goals [HaiHua and MiaoLiang, 2007].*

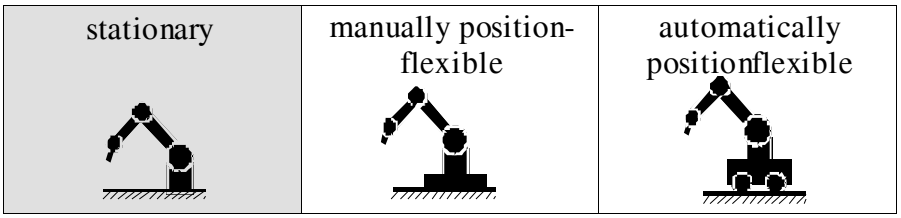


Figure 2.1: Stationary, manually positionflexible, and automatically positionflexible robot systems. Source: [Schraft et al., 2005].

Regarding the degrees of flexibility applicable to a robot system in general, the hardware system used and defined belongs to the automatically positionflexible robots as given in figure 2.1 [Schraft et al., 2005]. Figure 2.2 presents three further examples of industrial mobile robots that are automatically positionflexible.

(a) OnmiRob concept. (b) Mobile workstation. (c) Transportable robot.

Figure 2.2: Examples of automatically positionflexible mobile robots. Source: [Sprunk et al., 2011, Henkel & Roth, 2008, Bernhardt et al., 2008]

After an introduction in mobile robotics, the next sections concentrates on robot architectures in section 2.1.1, and their reconfigurability aspects in section 2.1.2.

2.1.1 Classical Mobile Robot Architectures

As the desired MobComm reconfiguration requires a suitable and supporting system architecture, the State of the Art in mobile robot architectures is reviewed in this section.

Research in mobile robots has focused almost exclusively on planning and world modelling in the sense plan act (SPA) approach until 1985. In the mid-1980s, Subsumption [Brooks, 1990] architecture was introduced as a departure from SPA. It is an attempt of Subsumption [Brooks, 1990] to make SPA more efficient by applying task-dependent constraints to the Subsumption layers.

In the beginning of the 1990s other control solutions for mobile robots arose with mainly three components: A reactive feedback control mechanism, a slow deliberative planner, and a sequencing mechanism that connects the first two components. One of these concepts is the 3T robot control architecture [Gat, 1992, Firby, 1989]. Besides the influential 3T and Saphira [Konolige and Myers, 1996], the MAS4AMR [HaiHua and Miao-Liang, 2007] approach is presented in the following.

3T

The three layer architecture [Gat, 1992], 3T for short, has been constituting the most influential mobile robot architecture since the 1990s, and consists of three architecture levels. The lowest level is a collection of soft real-time routines which can be rearranged into different control loops. The highest level, the plan execution system, manipulates the set of routines running to create a control sequence to accomplish a specific task. In skill layer, each skill is defined by a separate program and can be enabled dynamically. The activation of a skill follows its parametrisation and execution. The total system is a reactive plan interpreter. A set of task goals is taken as an input and each goal is refined hierarchically until primitive actions are reached. These hierarchical plans are called Reactive Action Packages (RAP) and they are located in the system library. A task plan must be able to create new tasks that are unconstrained by the current task refinement hierarchy, and to refer to and act on any task within the system. In 3T, sensor processing, learning, and world modelling are sparsely integrated [Gat, 1998].

This established robot architecture provides a basis for the MobComm architecture, presented in chapter 3, and allows for reliance on an established segmentation into three layers. Further, 3T implements the concept of dynamically selectable skills, relevant for

the configuration of the robot system. A hardware-abstracted handling of process changes (Objective 3) is beyond the scope of 3T.

Saphira

The mobile robot architecture Saphira [Konolige and Myers, 1996] has also been very influential since the 1990s. In contrast to 3T, Saphira focuses on world modelling and sensor processing and is an integrated sensing and control architecture for robotic applications with a behaviour-based control. The core part is its Local Perceptual Space (LPS), a geometric representation of the robot environment which provides an environment awareness for the robot. The reactive behaviours and action routines are on the action level of the architecture whereas perceptual routines are on the sensor level. The control of the system is accomplished by the procedural reasoning system. It coordinates task sequencing, system monitoring and perceptual coordination. Further, Saphira includes an optimised interaction between perception and action through the LPS and the advanced behaviour-based control. Complex tasks are decomposed in simple behaviours and can be handled more easily, but the debugging of the emerged behaviour must be conducted experimentally.

The relevance of this established robot architecture is its behaviour-based control approach, applying the basic modularity required for the implementation of reconfigurability in mobile robot systems. Its limit for MobComm is the lack of a hardware-abstracted functional reconfigurability (Objective 3).

MAS4AMR

Compared to 3T and Saphira, MAS4AMR (MAS for Auto Mobile Robot) [HaiHua and MiaoLiang, 2007] is an isolated approach based on agent technology, but contains the implementation of a separated data and signal stream for the encapsulation of reasoning as desired in MobComm.

MAS4AMR contains a communication middleware and event-driven agents that enable the robot to initiate actions adaptively to the dynamical changes in the environment. Depending on their functionality, different types of agents are used. Sensory agents get sensor data as input and output the description of the environment to the processing agents. The processing agents handle information from other agents. Interface agents and assistant agents are used to provide additional control capabilities. The core part

of MAS4AMR is the bulletinboard that is a receiver of system events for concluding the system status from these events. The current system status is the key for further agent interaction in the system. The self-organising architecture integrated into MAS4AMR is based on the separation of the data stream and the signal stream. The data stream consists of object data, and the signal stream is composed of event and system status information. Due to the given architecture, the robot is able to organise its agents in both hierarchical and heterarchical structures depending on the environmental input data.

MAS4AMR is relevant for MobComm, as it handles the implementation of self-organisation in mobile robot control by using agent technology, and it introduces different agent topologies to track the goals in the system. The limitation of MAS4AMR is its low maturity in the concept phase and the lack of application references.

After the presentation of 3T [Gat, 1992], Saphira [Konolige and Myers, 1996], and MAS4AMR [HaiHua and MiaoLiang, 2007] as classical robot architectures, the reconfigurability aspect in mobile robots is further focused on the next section.

2.1.2 Reconfigurability in Mobile Robots

By giving the definition of reconfigurability in the following, a basic understanding of the term is provided. Subsequently selected implementations in mobile robots are presented.

Definition 2.2 (Reconfigurability) *The reconfigurability of a system derives from the system's configurability [...]. Reconfigurations are later conversions and modifications of structure, functionality, capacity and technology by replacing, supplementing and removing discrete, autonomously operating components [Dashchenko, 2006].*

According to this definition, the Meta-Level Component approach [Edwards et al., 2009] and SHAGE [Kim et al., 2006] are presented as examples of reconfigurable robot architectures.

Meta-Level Component

Three fundamental activities of a robotic system are sensing, computation, and control. In the Meta-Level Component approach, these activities are mapped into the meta-level components that can be monitored, managed and adapted by the respective higher layer. Thus, this approach supports adaptive layered architectures of arbitrary depth, and the construction of adaptation plans on-the-fly [Edwards et al., 2009].

The software adaptation on-the-fly is triggered autonomically according to adaptation policies and executed according to adaptation plans. Adaptation policies specify when an adaptation is necessary and what the outcome of an adaptation should be. Adaptation plans specify the sequence of adaptation operations necessary to achieve that outcome. Meta-level components instantiate, configure, monitor, and deploy application-level components whereby they are architecturally aware, meaning that they have access to an internal representation of the current application architecture of the system [Edwards et al., 2009]. The Meta-Level Component approach handles reconfiguration activities based on component or total robot failures.

The segmentation in fixed implementation and flexible monitoring and configuration components provides the idea of a system segmentation to maintain productivity in MobComm (Objective 1). Even if a high level of reconfigurability is provided in the work of [Edwards et al., 2009] and self-organisation is implemented by the autonomic adaptation to policies, it is beyond the scope to react to functional changes (Objective 3).

Reconfigurable Three Layer Architecture

The Reconfigurable Three Layer Architecture [Sykes et al., 2008] builds upon the 3T architecture [Gat, 1992], and overcomes the limited responsiveness to low-level behaviours in 3T [Sykes et al., 2008].

The Reconfigurable Three Layer Architecture provides two mechanisms to deal with unexpected events in the environment. The first is the use of reactive plans allowing to recover from reaching unexpected states after performing an action. And the second, pictured in figure 2.3, is the ability to react to component failures due to a software bug or an environmental problem by selecting alternative components whenever possible.

The change management layer is capable of deriving appropriate configurations not considered by the user. Finally, if both mechanisms are insufficient for some problem, the system tries to replan [Sykes et al., 2008]. The work presented in [Sykes et al., 2008] points out that classical robot architectures can be enhanced with reconfigurability. The strength of this approach is the dynamic reaction to software failures or environmental problems. Just like the formerly introduced architectures in this section, the reaction to functional process changes is not evaluated in this approach (Objective 3).

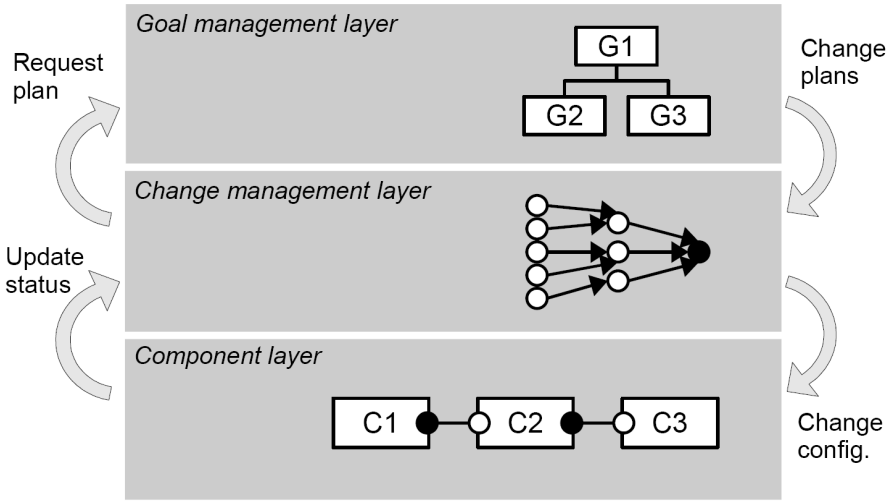


Figure 2.3: Reconfigurable Three Layer Architecture. Source: [Sykes et al., 2008].

SHAGE

The level of reconfigurability in SHAGE [Kim et al., 2006] (Self-Healing, Adaptive, and Growing Software) architecture is based on abstract and concrete system levels. At the abstract level only slots are comprised that represent abstract components describing services. Such a service outlines a functionality offered to the system. A service does not indicate a concrete component but describes the message a slot requests and the result it returns.

As opposed to the services in the abstract level, each slot is called by a concrete component in the form of an executable code at the concrete level. Every component in the framework must implement common and specific interfaces, containing messages that the component can receive and send.

The high level of reconfigurability is reached in SHAGE by using a Reconfigurator measuring mismatches between components and finding suitable connectors. In case two components were deployed in two different machines, the Reconfigurator selects a remote connector which enables remote communication and reconnects these components. The Reconfigurator sends a starting message to every new component in the architecture and reports the termination of reconfiguration to the architecture broker [Kim et al., 2006].

The SHAGE architecture is powerful concerning its message-based reconfigurability segmented into an abstract and a concrete level. The limitation of SHAGE lies in the missing reconfiguration after functional changes (Objective 3), and the missing division into productive and reconfiguration activities for a maintenance of productivity (Objective

1).

The presentation of SHAGE [Kim et al., 2006] concludes the review of reconfigurable robot architectures. After reviewing the domain-independent verification of hybrid computational systems, these approaches are discussed in section 2.1.4 concerning their relevance for MobComm.

2.1.3 Reconfiguration and Verification of Hybrid Systems

After the presentation of classical and reconfigurable mobile robot architectures, this section covers reconfiguration and verification aspects of hybrid computational systems. Hybrid systems "combine both digital and analogue components" [Alur et al., 2000] as given in reconfigurable robot control architectures. The task execution of a mobile robot is for example modelled as discrete events while the low level control follows a continuous behaviour. Hybrid system theory is applied to the area of mobile robots besides its use in several domains such as automated highway systems, air-traffic management systems, or embedded automotive controllers [Alur et al., 2000]. For the further use of this term, it is defined in the following:

Definition 2.3 (Hybrid system theory) *Hybrid system theory is the modelling, analysis, and control of systems which involve the interaction of both discrete state systems, represented by finite automata, and continuous state dynamics, represented by differential equations. [Tomlin et al., 2003]*

The application of hybrid system theory combines computer science and engineering control theory with the goal to design verification techniques for computational systems [Tomlin et al., 2003]. The application of the resulting techniques allows to both reconfigure and verify computational systems domain-independent.

Especially the need to verify behaviour of safety critical system components as for example the flight controller of aircraft is in the focus of this research area. The problem of safety verification is described as "the encoding of the condition of the region of operation in the system's state space" [Tomlin et al., 2003]. This includes that the states reachable from some initial set are very difficult to represent for mobile robots as continuous dynamics are involved at their low level behaviours.

Literature gives a large number of mechanisms for the modelling and reconfiguration of hybrid systems in different domains. Due to its domain-independent use and its appli-

cability to mobile robots, this review focuses on Linear Temporal Logic (LTL).

Linear Temporal Logic is able to provide a mathematical framework to express high-level tasks or environment specifications as temporal constraints. These constraints constitute the input of a formal verification algorithm. The corresponding formulae combine propositions with boolean $\neg$, $\vee$, $\wedge$ and temporal connectors such as X (next), U (until), R (release), F (future), G (globally) [Plaku, 2008].

An example application of Linear Temporal Logic is the transformation of an expression in natural language into a mathematical formula:
"After inspecting a contaminated area A, visit a decontamination station B, before returning to any of the base stations C or D."
The following Linear Temporal Logic results [Plaku, 2008]:

$$F\left(\pi_A \wedge \left(\left(\neg\pi_C \wedge \neg\pi_D\right) U\left(\pi_B \wedge F\left(\pi_C \vee \pi_D\right)\right)\right)\right)$$

Temporal logic specifications as given above can capture the traditional control specifications such as reachability and invariants (cf. figure 2.5) as well as more complex specifications like sequencing and obstacle avoidance [Fainekos et al., 2009]. This expressiveness is one reason for its powerful and accepted use in hybrid systems. Another reason for its relevance for MobComm is the resemblance of the used logic to natural language. These prospects are closely related to the semantic integration of new robot functionalities in MobComm.

After the general introduction to hybrid system control and to Linear Temporal Logic an approach for its application in motion planning [Fainekos et al., 2009] is presented in the following.

Temporal Logic Motion Planning for Dynamic Robots

The "main challenge in robotics" [Fainekos et al., 2009] to develop the mathematical framework to formally and verifiable integrate high level planning with continuous control primitives is handled in the approach of [Fainekos et al., 2009]. The motion planning follows a hierarchical approach that designs control laws for a fully actuated kinematic model of the robot.

The solution resulting from these control laws guarantees to satisfy the initial user specification which is closely related to the dependability of the reconfigured skill in MobComm

as desired in Objective 2.

The computational sequence of the motion planer in [Fainekos et al., 2009] can be divided into three steps:

1. Discrete abstraction of robot motion: The environment is decomposed into a finite number of equivalent classes as presented in figure 2.4 with the convex cell composition of the robot workspace.

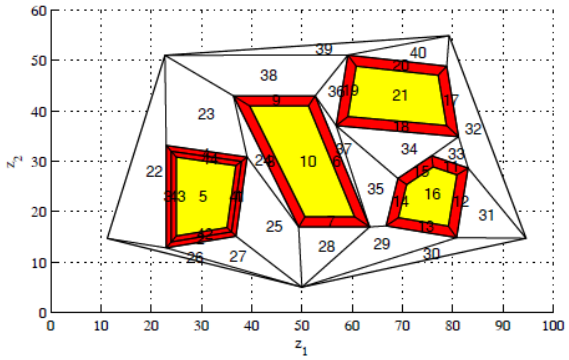


Figure 2.4: Discrete abstraction of robot motion: Example of a convex cell composition. Source: [Fainekos et al., 2009]

2. Temporal logic planning: This step includes the construction of plans for discrete robot motions which have to satisfy given requirements (as expressed in temporal logic). By using automata theory, the design of the hybrid controller for the motion planning is executed.

3. Continuous implementation of discrete plans: The focus in this step is the implementation of the continuous feedback control laws *reachability* and *invariant controller*. The *reachability* controller (left side of figure 2.5) is responsible to drive each state inside a cell to a predefined boundary of the cell while the *invariant controller* (right side of figure 2.5) guarantees that all trajectories that start inside a cell will remain safely in that cell. The illustration of the two feedback control laws is overviewed in figure 2.5

The relevance of hybrid system control by using Linear Temporal Logic for this work is in close association with the resemblance to natural language. Further, the powerful modelling of continuous control laws and the description of tasks as discrete events allow

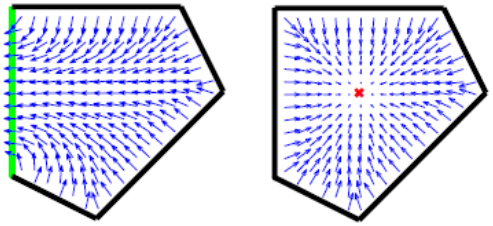


Figure 2.5: Application of feedback control loops: Reachability and cell invariant controller. Source: [Fainekos et al., 2009]

to dynamically change and formally verify system behaviours. Combined with the guarantee to fulfil a user-inserted specification this research area is able to comply with the dependability requirement of Objective 2.

Due to the key influence of low level modelling on the hybrid controller, the required hardware-abstraction cannot be complied with hybrid system theory (Objective 3).

2.1.4 Discussion

A wide range of different mobile robot control architectures have been investigated for the last thirty years. In the last decade reconfigurability in mobile robot systems has also been focused on. Classical robot architectures and approaches focussing on reconfigurability are given in section 2.1.1 and 2.1.2. Even though the individual approaches have been evaluated regarding their relevance for this work directly after their explanation, table 2.1 gives a summary of the results and exposes the degree of objective fulfilment. As the first and the third research objectives contain two aspects, a separated analysis is made for the evaluation in table 2.1.

Starting with the self-organisational aspect of Objective 1, some approaches comply with this characteristic. SHAGE [Kim et al., 2006] provides self-organising capabilities by its message-based reconfigurability, MAS4AMR [HaiHua and MiaoLiang, 2007] applies agent technology, and the Meta-Level Components [Edwards et al., 2009] adapt policies to implement self-organisation.

The architectural segmentation in the Meta-Level Components [Edwards et al., 2009] and SHAGE [Kim et al., 2006] does not fully provide a solution for the maintenance of productivity required in the second aspect of Objective 1 but gives a basic idea about a possible solution in MobComm. In contrast, the implementation of dependability (Objec-

	Self-organisation (Obj. 1)	Maintenance of productivity (Obj. 1)	Dependability (Obj. 2)	Functional changes (Obj. 3)	Hardware abstraction (Obj. 3)	Characteristics	
3T [Firby, 1996]	No	n/a	n/a	n/a	No	- Established segmentation into three layers.	- Functional reconfiguration aspects not regarded.
Saphira [Konolige and Myers, 1996]	No	n/a	n/a	n/a	No	- Implementation and reconfiguration components given.	
MAS4MAR [HaiHua and MiaoLiang, 2007]	Yes	n/a	n/a	n/a	n/a	- Use of an inner and outer part of the architecture to allow self-organisation. - Lack of evaluation and implementation	
Meta-Level Components [Edwards et al., 2009]	Yes	part.	n/a	No	n/a	- Division into reconfiguration and task execution in different layers.	
Reconfigurable 3T [Sykes et al., 2008]	Yes	n/a	n/a	No	n/a	-Reconfigurability after software and hardware breakdowns is given. - Put on classical 3T architecture	
SHAGE [Kim et al., 2006]	Yes	part.	n/a	No	n/a	- Message-based reconfigurability segmented into abstract and concrete architecture levels.	
Temporal Logic Motion Planning [Fainekos et al., 2009]	n/a	n/a	Yes	n/a	No	- Guaranteed fulfilment of user-inserted specification by the provision of a formal framework using Linear Temporal Logic.	

Key: No = Objective not fulfilled, part. = Objective partly fulfilled, Yes = Objective fulfilled,
n/a = No statement about objective available.

Table 2.1: Summary of level of objective fulfilment and characteristics of presented mobile robot architectures.

tive 2) is not mentioned in any of the classical or reconfigurable robot control approaches. The verification of hybrid computational systems as proposed in [Fainekos et al., 2009] is able to comply with this requirement by providing a formal mathematical framework.

The required reaction to functional changes, as one part of Objective 3, is only evaluable for the reconfigurable robot architectures, but not in the scope of Meta-Level Components [Edwards et al., 2009], Reconfigurable 3T [Sykes et al., 2008], and SHAGE [Kim and Robertazzi, 2006]. The second part of Objective 3, the required hardware abstraction, is not a goal of the classical robot control architectures like 3T [Firby, 1996] or Saphira [Konolige and Myers, 1996] as the control directly relies on the hardware capabilities of the robot. Even if the reconfigurable approaches do not completely rely on the

hardware, its abstraction, as desired in Objective 3, is not part of the evaluated characteristics.

The relevance of the mobile robotics research area for MobComm lies in its established segmentation of robot architectures as given in 3T [Firby, 1996], and the solution idea to allow self-organisation (e.g. Meta-Level Components [Edwards et al., 2009]) and maintenance of productivity (e.g. SHAGE [Kim et al., 2006]). Due to the focus on mobile robot control and the missing integration of functional reconfigurability, none of the presented approaches completely fulfils the given research objectives in this area. The same set of objectives is reviewed in the next section regarding approaches provided by the manufacturing system area.

2.2 Manufacturing Systems

After the presentation of related work concerning mobile robotics in section 2.1, this review gives details of the manufacturing systems area. The set of MobComm objectives is closely related to investigated issues in manufacturing systems. Even if the field of application in MobComm differs from the scope of manufacturing systems, reconfigurability, hardware abstraction, and dependability are key goals in manufacturing and thus match with the MobComm objectives.

This review starts with an abstract model of a manufacturing system that consists of handling, machinery, labour, or knowledge among others given in figure 2.6. The scope of MobComm is the machinery part of a total manufacturing system only, as the used mobile robot describes a single manufacturing component.

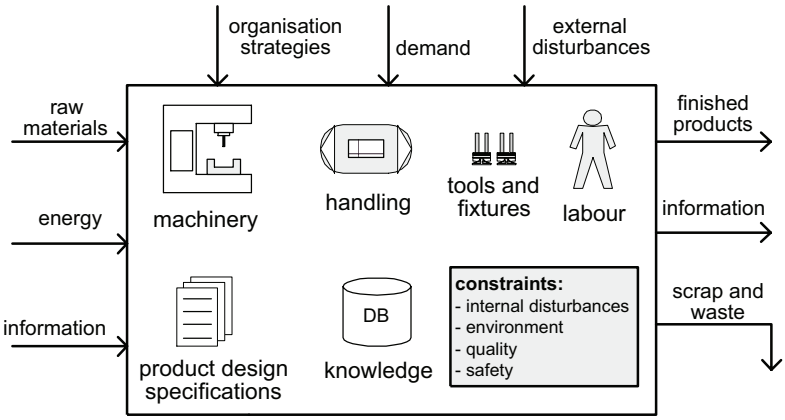


Figure 2.6: Abstract model of a general manufacturing system. Source: [Leitão, 2004].

Mechanisms applied to implement flexibility and reconfigurability in manufacturing systems are examined in the following to be adapted to the reconfiguration of industrial mobile robots. For a clarification of the terms manufacturing system, manufacturing component, and manufacturing control in this work, their definitions are given in the following:

Definition 2.4 (Manufacturing System) *Manufacturing systems involve activities related to the production of goods using manufacturing resources and knowledge, according to the external demands and subject to the environmental context, e.g. social and economic aspects [Leitão, 2004].*

Definition 2.5 (Manufacturing Control) *Manufacturing control handles the internal logistics in a manufacturing system. It decides about all the routines of product instances as well as the starting of production processes on these unfinished products [Valckenears et al., 2001].*

Definition 2.6 (Manufacturing Component) *A manufacturing component is a physical equipment that can perform a set of specific functions [...]. It is able to execute one or more basic production actions, e.g. moving, transforming, fixing or grabbing [Barata and Camarinha-Matos, 2003].*

In the context of a complete manufacturing system the robot used in MobComm is a manufacturing component, controlled by the corresponding manufacturing control. Robot-internal activities, such as the reconfiguration mechanism investigated in this work, are not regulated by the manufacturing control any more. Even if the field of application in this thesis goes beyond the scope of established manufacturing control, the objectives in this work are similar to those given in the report "Visionary Manufacturing Challenges for 2020" [Committee on Visionary Manufacturing Challenges, 1998]. This report states the achievement of concurrency in all operations, and a rapid reconfiguration of manufacturing enterprises in response to changing needs to name just some examples amongst a list of others.

As objectives of MobComm correlate with those of manufacturing systems, the following section elaborates different manufacturing paradigms. After the presentation of self-organisation in manufacturing systems in section 2.2.2, the review of knowledge engineering in manufacturing is the focus of section 2.2.3.

2.2.1 Manufacturing Paradigms

During the past century different manufacturing paradigms were applied in industry to increase competitiveness. Not all of them are relevant to this work, as only a small set explicitly supports reconfigurability, self-organisation, or flexibility.

Figure 2.7 gives an overview of the most established manufacturing paradigms, assigned to the principle of mass production and mass customisation. While mass customisation is shaped by the customer's demand for specific and customised products, mass production has no or a low variety in the resulting products [Leitão, 2004]. Therefore, the variety of the resulting product lays the foundation for the classification of manufacturing paradigms in figure 2.7.

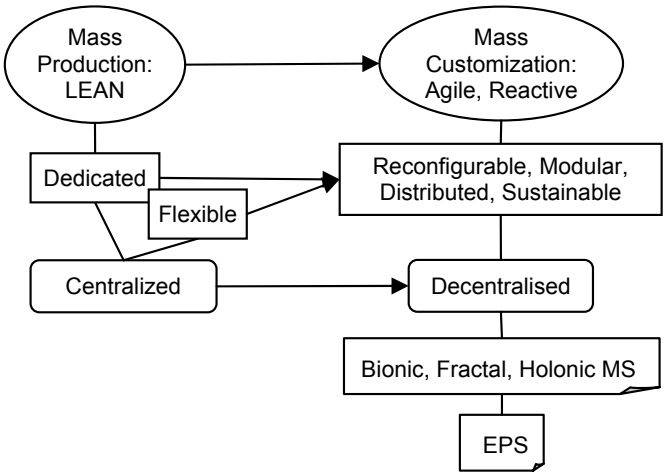


Figure 2.7: Classification of different manufacturing paradigms. Source: [Frei et al., 2007a].

This section reviews flexible manufacturing systems (FMS), reconfigurable manufacturing systems (RMS), agent-based manufacturing systems, and finally holonic manufacturing systems (HMS) as they are all related to the required reconfigurability, self-organisation, and flexibility. Due to its strong self-organising aspect, evolvable assembly systems (EAS) are presented in section 2.2.2 along with self-organisation in manufacturing systems.

Flexible and Reconfigurable Manufacturing Systems

Following figure 2.7, reconfigurable and flexible manufacturing systems are based on a centralised control structure. As a consequence, the key idea of FMS is the coordination of the work flow, carried out by a centrally-controlled computer [Upton, 1992]. The following functions shape a FMS:

- Scheduling of jobs on the machine tools,
- download of part-programs to machines, and
- sending of instructions to the automated vehicle system for part transportation [Upton, 1992]

Based on this central control structure, the following definition describes a FMS:

Definition 2.7 (Flexible Manufacturing System) *A flexible manufacturing system is a machining system configuration with fixed hardware and fixed, but programmable, software to handle changes in work orders, production schedules, part-programs, and tooling for several types of parts [Mehrabi et al., 2002].*

A FMS has an examplary layout as given in figure 2.8, and consists of a set of work stations, interconnected by a transport and material handling system, controlled by an integrated control system. Following [Mehrabi et al., 2002], flexible manufacturing is still part of the CNC (Computer Numerically Controlled) epoch, whereas Reconfigurable Manufacturing Systems are already assigned to the knowledge epoch in manufacturing starting in the 1990s.

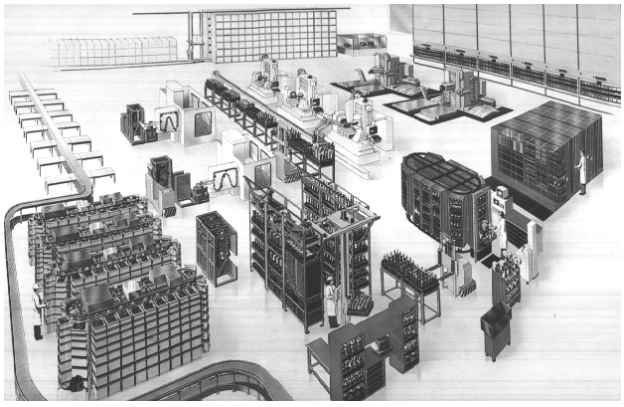


Figure 2.8: Example layout of a FMS. Source: [Leitão, 2004].

Changing from mass production in FMS to mass customisation, as given in car manufacturing, the flexibility paradigm is further developed to the reconfigurability paradigm in RMS [Barata et al., 2005]. RMS are machining systems that can be replaced quickly and reliably [Mehrabi et al., 2002], and follow the definition given below.

Definition 2.8 (Reconfigurable Manufacturing System) *A reconfigurable manufacturing system is defined as a system [...] for rapid change in structure, as well as*

in hardware and software components, in order to quickly adjust production capacity and functionality within a part family [Koren et al., 1999].

In contrast to FMS, RMS include modularity, convertibility, and customisation as their key characteristics. As the original RMS is still based on a central control, characteristics like modularity or integrability are sparsely developed compared to its sub-type, the agent-based manufacturing paradigm. These decentralised control systems are the focus of the next section that describes general characteristics and implementations.

Agent-Based Manufacturing Systems

Following [Shen and Norrie, 1999], agent-based manufacturing systems are divided into three categories: Enterprise integration and supply chain management, manufacturing scheduling and control, and holonic manufacturing. The holonic approach, reviewed in the next section, is preceded by the manufacturing scheduling and control in this section.

For clarification, well-established definitions of the terms agent and multi-agent system (MAS) are provided:

Definition 2.9 (Agent) *An agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous actions in that environment to meet its design objectives [Jennings, 1999].*

Definition 2.10 (Multi Agent System) *A MAS is a loosely coupled network of agents, according to definition 2.9 that interacts to solve problems that are beyond the individual capabilities or knowledge of each problem solver [Durfee and Lesser, 1989].*

The software entities contained in this paradigm are agents forming a MAS. Even though agent technology is the basis for more flexible manufacturing, only few applications can be found in real automation systems according to [Leitão and Restivo, 2008]. A number of reasons restrain companies from using production-related agent technology. Agent applications are confined to isolated applications until they are in a wide-spread use. Consequently, existing industrial standards cannot be used in single applications, and new maintenance standards that occur generate high follow-up costs for companies.

Despite these barriers to use agent technology in industry evaluated in literature such as [Leitão, 2004,Wagner, 2002], the agent roadmap about the future use of agents constitutes that manufacturing will be one of the most significant domains for this technology [Luck

et al., 2005]. Figure 2.9 provides the results of a survey about agent use, and indicates the number of times the manufacturing domain was selected as influential for this technology in the future by experts.

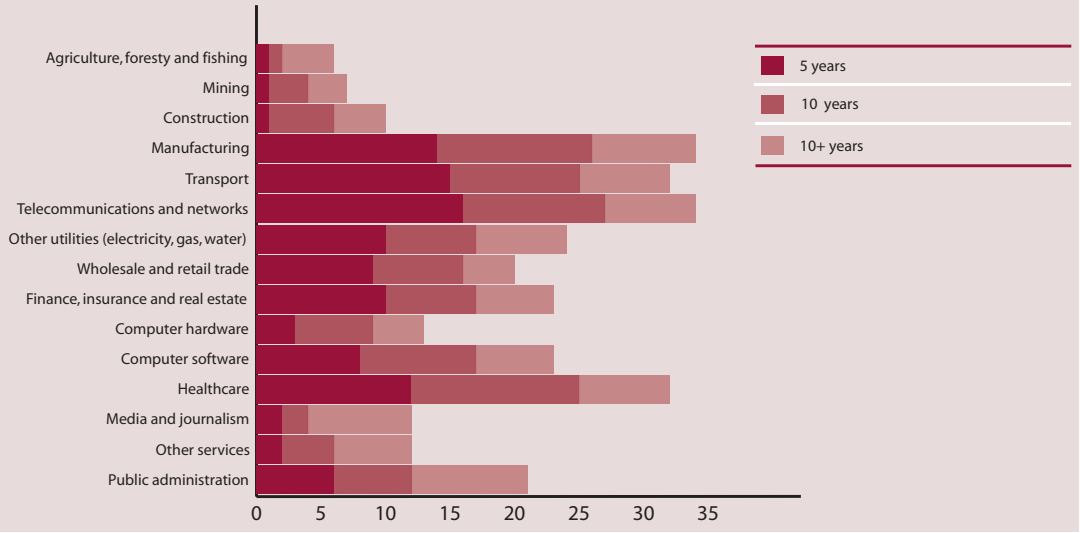


Figure 2.9: Survey results of significant domains for future agent use. Source: [Luck et al., 2005].

The reasons behind the good performance of manufacturing in the survey are given in [Pěchouček and Mařík, 2008]. The main capabilities expected of agent-based solutions for the manufacturing industry are the robustness of highly distributed solutions, the capability of replanning operations on the fly, and a simple way of extending both the hardware and software when modifying the manufacturing equipment.

These indicated advantages for manufacturing are in contrast to the missing flexibility of hierarchical control systems that are traditionally arranged by programmable logic controllers (PLC). There, different functional software levels are given according to the hardware structure whereas the hardware devices are modelled as predefined software components that have to be configured and connected in every application.

The distributed approaches, resulting from agent use, have the important advantage of low complexity in the control software. Further, they are modular and, by definition, show emergent robustness when facing disturbances, component failure, or other critical situations, as desired in MobComm. One of the first applications of agent technology in manufacturing is presented in [Bussmann and Schild, 2000], and aiming at making the process of manufacturing more flexible and robust additionally to the demonstration of enhanced scalability of the manufacturing system. In contrast to traditional manufacturing

solutions, no central control unit is used any more in the approach of [Bussmann and Schild, 2000].

A selected set of more recent agent-based approaches is presented in the following. CoBASA [Barata and Camarinha-Matos, 2003], Agent-based Commissioning [Staab et al., 2004], and PABADIS [Klostermeyer and Klemm, 2003] are discussed concerning their relevance for MobComm.

CoBASA

An approach to dynamically re-engineer shop floor controls is presented in [Barata and Camarinha-Matos, 2003]. The Coalition Based Approach for Shop Floor Agility (CoBASA) deals with planned orders in the shop floor control leading to a set of outputs. In CoBASA a re-engineering phase is introduced to facilitate the handling of changes in the control structure. The main objective of CoBASA is the support of a fast adaptation to changes with minimal effort [Barata and Camarinha-Matos, 2003].

The basic components of CoBASA are manufacturing components, manufacturing resource agents (MRA) , coordinating agents (CA), clusters, coalitions, brokers, and contracts. Coalitions are the CoBASA key characteristic and composed of agentified manufacturing components whose relationships are dominated by contracts configured whenever a coalition is established [Barata and Camarinha-Matos, 2003]. The creation of new coalitions does not require programming efforts as the used contracts are able to handle changes dynamically. Figure 2.10 gives two CoBASA coalitions, with the first coalition not having direct access to members of the second one.

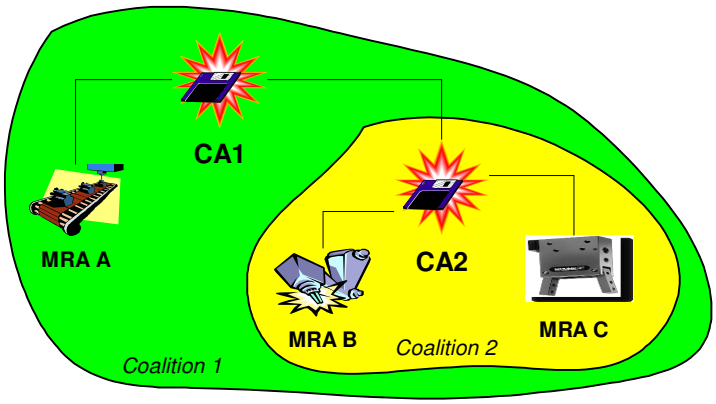


Figure 2.10: Example of CoBASA coalitions. Source: [Barata and Camarinha-Matos, 2003].

Agents contained in the coalitions have generic capabilities in form of agent skills, whereas the according tasks are regulated by the flexible contracts. These contracts allow different structures following different goals while using the same agents. Changes made in the capability of an agent are propagated to the rest of the architecture via changed contracts. The separation of agent competences, implemented as skills, and agent cooperation, regulated by contracts, facilitates the development of very agile manufacturing control structures.

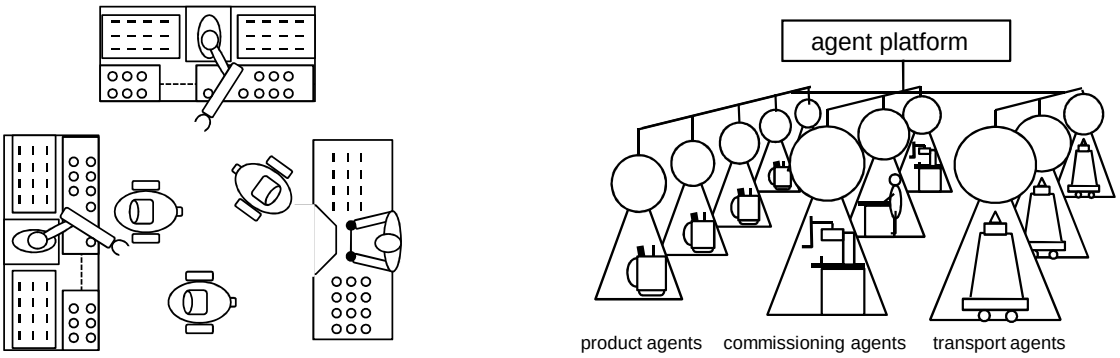
The relevance of CoBASA for this thesis is the re-engineering of existing structures due to new system objectives. High level skills that are offered to coalitions by composing basic skills provide a basic for the self-organised reconfigurability in MobComm (Objective 1). Contrary to this relevance, the scope of this approach does not include a hardware-abstracted reaction to functional changes.

Agent-based Commissioning

Compared to CoBASA, the agent-based commissioning correlates with the field of application given in MobComm. In [Staab et al., 2004] a MAS for industrial commissioning as described in the introduction section 1.1 is investigated. Even though the commissioning process is based on distributed hardware presented in the layout of figure 2.11(a), comparability to the commissioning processes in this work is given. Commissioning is a relevant field of agent use, as it is an arbitrary process in manufacturing, and hard to be programmed in traditional ways. The complexity of producing and transporting different products in parallel is very high. Especially the enhancement of commissioning systems with new products requires a costly and error-prone programming in traditional concepts [Staab et al., 2004]. These disadvantages are mitigated by the use of agents. The commissioning scenario in [Staab et al., 2004] is composed of stationary industrial robots, automated guided vehicles (AGV), and human workers.

Figure 2.11(b) presents the MAS including the agent platform as its central unit. The used platform provides common services and data, manages all agents, and additionally stores information about communication interfaces. Three different types of agents map the commissioning process. Product agents represent the actual product, commissioning agents act as agentified manipulators, and transport agents handle the connecting AGV. The agent interaction is implemented by Jini technology [Staab et al., 2004].

As mentioned above, the agent-based commissioning approach in [Staab et al., 2004]



(a) Top view on the commissioning scenario.

(b) System structure and agent types.

Figure 2.11: Overview and structure of the Agent-based Commissioning approach. Adapted from: [Staab et al., 2004].

resembles MobComm in the field of application, and provides a distributed solution regarding the hardware for a flexible commissioning process of different products. The limitation of this approach regarding MobComm objectives is the missing reconfigurability after functional process changes (Objective 3), as the commissioning process itself is regarded as static.

PABADIS

Less focussing on a specific field of application than the last approach, the Plant Automation Based on Distributed System (PABADIS) approach [Klostermeyer and Klemm, 2003] and its extension Product Oriented Manufacturing Systems for Reconfigurable Enterprises (PABADIS'PROMISE) [Peschke et al., 2005] are results of European research projects, and aiming at improving flexibility in manufacturing by extending the application of distributed intelligence. In the following, the basic PABADIS and PABADIS'PROMISE architectures are presented.

Traditionally, layers in manufacturing, as given in figure 2.12, implement the Enterprise Resource Planning (ERP), the Manufacturing Execution System (MES), and on the bottom, the field control layer strictly separate. This compelling separation is increasingly suspended with PABADIS. By using agent technology, transitions between office, factory, and field level can be modelled more modular and flexible.

PABADIS is a distributed, resource-centred control architecture with parametrisation on demand. The limitation of this manufacturing architecture towards MobComm lies in the missing dynamic between the factory and the field level [Peschke et al., 2005]. Due to

this limitation, a more modular and flexible architecture was build in PABADIS'PROMISE. Hence, PABADIS'PROMISE is able to maximise product and process flexibility between all given layers as presented in figure 2.12.

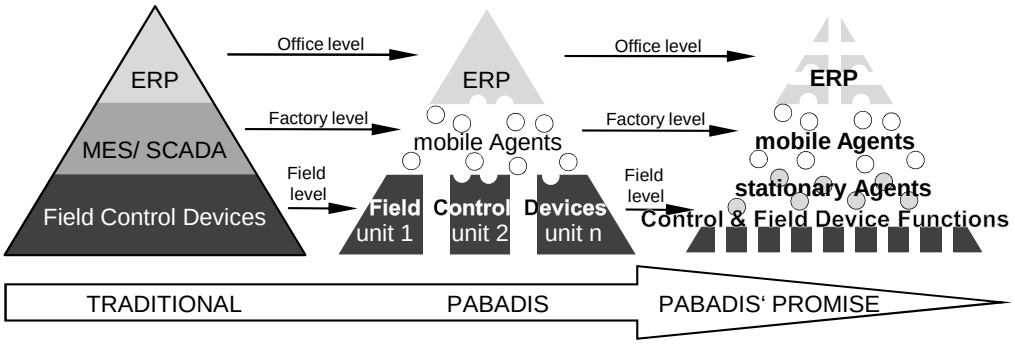


Figure 2.12: Progression from traditional structures to PABADIS and PABADIS'PROMISE layers. Adapted from: [Peschke et al., 2005].

Application efficiency is improved by the implementation of order-related manufacturing, and flexibility is gained by shifting the decision-making process, formerly located in the ERP, down to the MES layer, as presented in figure 2.13 [Feng et al., 2007].

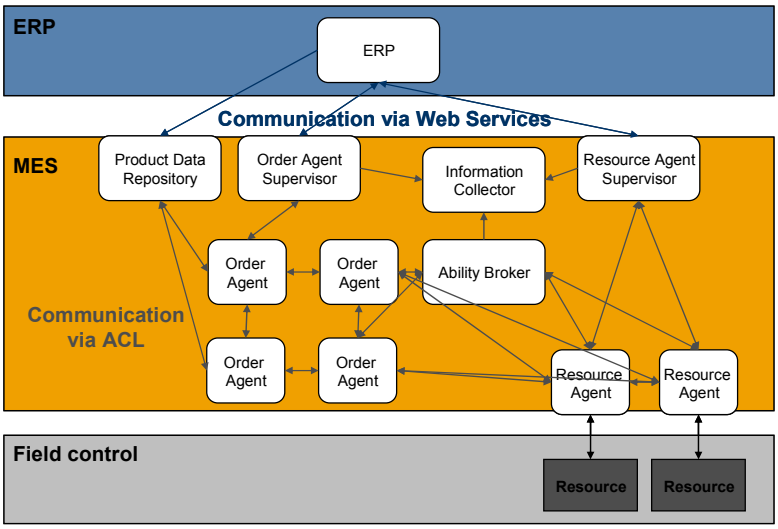


Figure 2.13: Overview of PABADIS'PROMISE architecture. Source: [Feng et al., 2007].

The ERP interface is responsible for all actions related to the order management whereas the MES kernel contains the set of order agents. For the interface to the field control layer a set of resource agents is provided. The flexibility of integrating dynamically changing products is increased in PABADIS'PROMISE by using a modular inner-agent communication structure and protocols.

PABADIS'PROMISE is relevant for this work as different manufacturing layers are connected dynamically and agent-based. PABADIS'PROMISE shifts the decision making in manufacturing control from the ERP layer to the MES layer, likewise MobComm desires to shift reconfigurability from a hardware controlled mechanism to a hardware-abstracted application layer of an industrial mobile robot. The agentification of the field layer provides hardware abstraction as desired in MobComm (Objective 3) that can be found in all approaches following the agent-based manufacturing paradigm. The relevance for MobComm faces the limitation that PABADIS'PROMISE focuses on order decomposition contrary to the desired functional reconfigurability in MobComm (Objective 3).

Following the comparison in [Lueder et al., 2005] between PABADIS and the holonic paradigm, as well a sub-type of RMS, many goals connect the presented approach and the holonic paradigm, as elaborated in the following section.

Holonic Manufacturing Systems

The research area of holonic manufacturing systems (HMS) was initiated in 1987 when the first approach of heterarchical-oriented control structures in manufacturing was introduced by [Duffie and Piper, 1987]. For a deeper historical background of HMS the reader is referred to [Colombo et al., 2005]. Today many applications of HMSs can be found in literature as shown in [Leitao, 2009].

The holonic manufacturing is also motivated by the need of increased manufacturing flexibility. To reach this flexibility, HMS take advantage of the holonic paradigm. Holons have two important characteristics: the ability to act autonomously and to cooperate with other holons to transform into effective components of bigger wholes [Leitao, 2009]. Even though a holon is presented as an independent concept in literature, it is based on agent technology. The relation between holons and agents is more extensively worked out in [Giret and Botti, 2004]. The most unique characteristic of holons is their recursiveness, and leads to the following definition:

Definition 2.11 (Holon) *A holon is an autonomous and co-operative building block of a manufacturing system for transforming, transporting, storing and/or validating information and physical objects. The holon consists of an information processing part and often a physical processing part. A holon can be part of another holon [Van Brussel et al., 1998].*

The holarchy, a cluster of holons, is the model of a generic self-organising structure in

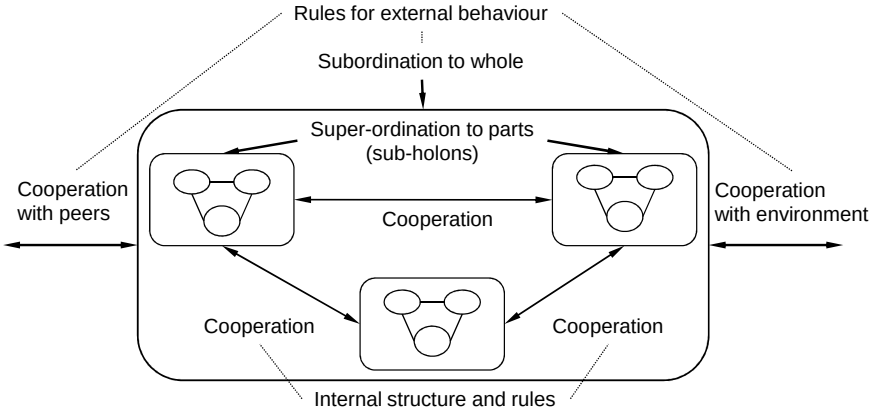


Figure 2.14: Generic model of a holarchy. Adapted from: [Ulieru, 2004].

which the holons behave as autonomous wholes and cooperative parts for achieving the goal of the holarchy at several levels [Ulieru, 2004]. This representation of a holarchy is illustrated in figure 2.14, and emphasises the key aspect of holonic autonomy while being part of a bigger whole.

After defining a single holon and describing the resulting holarchy, figure 2.15 presents the model of a complete holonic system. Following [Colombo et al., 2005], the basis for every HMS is the agentification of the given resources. Thus, each manufacturing component is mapped into an agent that abstracts all parameters of the component needed for its control. The agent-based communication capabilities transform the manufacturing component into a self-reconfiguring intelligent element, the holon [Colombo et al., 2005]. Consequently, a cluster of holonic components form a HMS as presented in figure 2.15. The HMS Consortium has defined more generalised characteristics of a HMS: Autonomy, Cooperation, and Openness [Giret and Botti, 2004].

After the introduction of holons, holarchies, and the resulting HMS, PROSA [Van Brussel et al., 1998] and ADACOR [Leitão and Restivo, 2008] are given in the following as examples of established holonic manufacturing architectures.

PROSA

The Product-Resource-Order-Staff Architecture (PROSA) [Van Brussel et al., 1998], presented as the most established representative of holonic manufacturing (e.g. in [Lueder et al., 2005]), applies the holonic manufacturing principle. As PROSA is a holonic architecture, literature gives a set of concrete manufacturing applications implemented with PROSA like MASCADA (Manufacturing Control Systems Capable of Managing

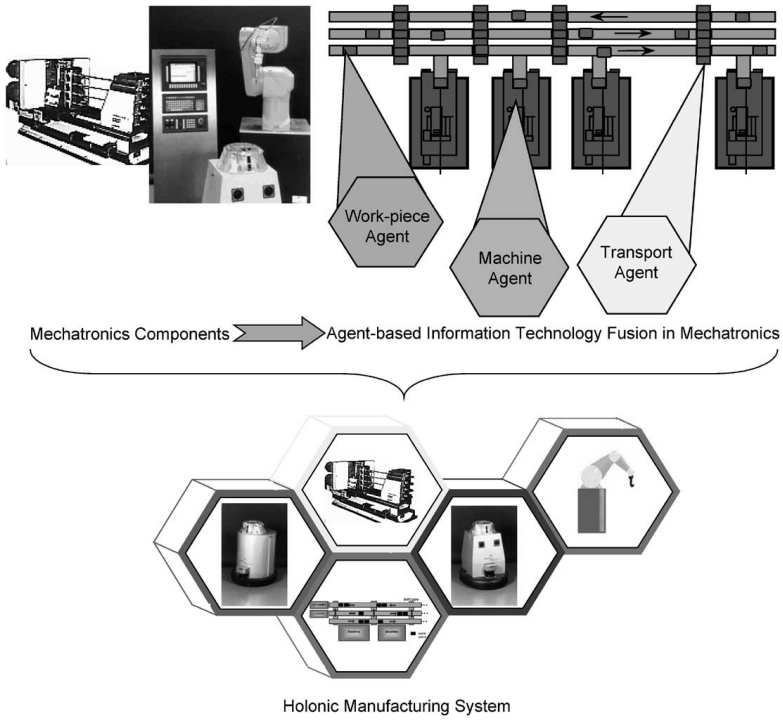


Figure 2.15: General model of a HMS. Source: [Colombo et al., 2005].

Production Change and Disturbances) [Kollingbaum et al., 2000, Babiceanu and Chen, 2006].

Since this holonic architecture complies with the object-oriented paradigm, data does not exist on its own but always belongs to the holon that maintains it. For example, the resource holon maintains data of its capabilities, tasks, resources, and activities. Order and product holons hold corresponding data about the manufacturing order and the desired product. The process, its execution, and the production knowledge is exchanged bidirectionally between different types of holons. All types of holons including the data exchange are pictured in figure 2.16.

A characteristic of PROSA is the self-similarity of the holons that determines the level of reconfigurability of the control system. The homogeneous system components reduce the complexity of the overall system and simplify the development and integration of new holons into the system. PROSA contains holons of the same type with similar behaviour and interfaces. This horizontal self-similarity allows the holons to be internally different while not imposing additional complexity to other holons. The according vertical self-similarity is the "whole" that becomes the "part" of the bigger "entity" [Van Brussel et al., 1998].

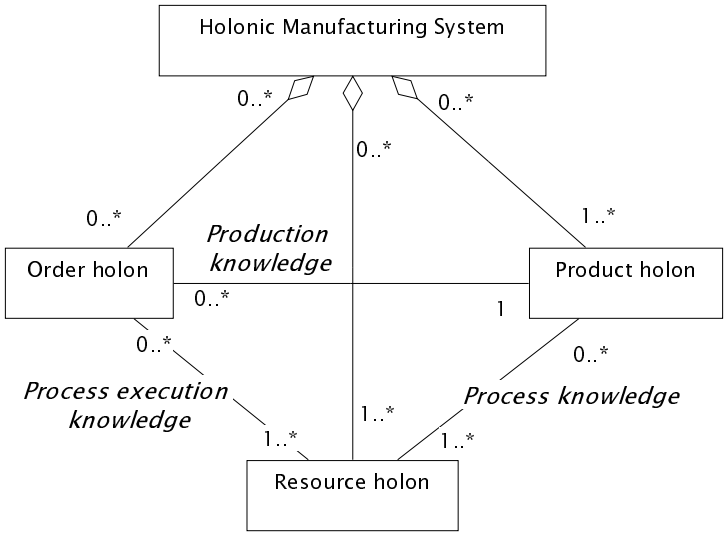


Figure 2.16: Overview of PROSA holons and data exchanges. Source: [Van Brussel et al., 1998].

As the system structure of PROSA is decoupled from the control algorithm, PROSA allows incorporating hybrid control algorithms. Hybrid algorithms neither follow completely a heterarchical nor a hierarchical structure as explained in figure 2.17. Depending on the system requirements, both control principles can be applied dynamically in a holonic architecture.

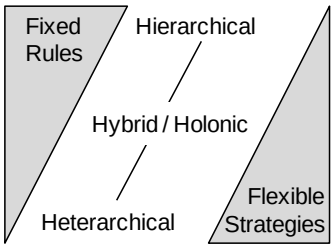


Figure 2.17: Comparison of hierarchical, hybrid, and heterarchical control structures. Adapted from: [Bongaerts et al., 2000].

PROSA is relevant to MobComm as its hybrid architecture provides the basic capability to implement a self-organised reconfiguration combined with the maintenance of productivity (Objective 1). The distinctive scope of PROSA, the control of a complete manufacturing system, and the missing reconfigurability after functional changes are its limitations towards the given research objectives.

A more recent approach of a HMS is given in ADACOR [Leitão and Restivo, 2008]. Even if PROSA and ADACOR [Leitão and Restivo, 2008] share a set of common goals,

the next section emphasises the unique characteristics of both examples.

ADACOR

The Adaptive Holonic Control Architecture (ADACOR) is also based on the holonic principle. Product, task, and operational holons resemble the concepts used in PROSA [Van Brussel et al., 1998], in contrast to the supervisor holon that outlines a unique characteristic of ADACOR. It provides the possibility to coordinate other supervisor holons in a federation architecture. This coordination allows managing the dynamic evolution of holarchies due to the environmental context [Leitão, 2004].

Product holon (PH), task holon (TH), operational holon (OH), and supervisor holon shape the resulting architecture. An overview of ADACOR architecture is presented in figure 2.18.

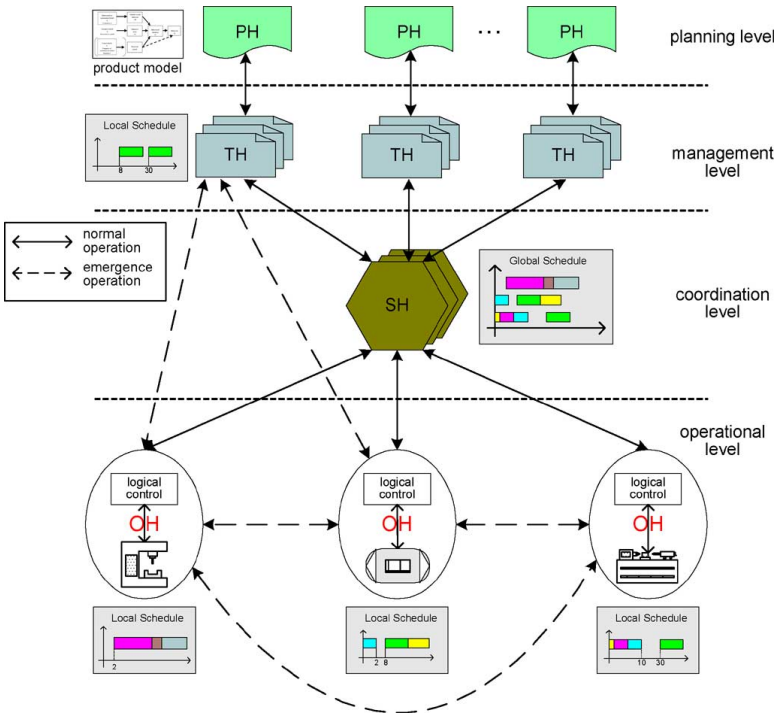


Figure 2.18: Overview of ADACOR holons and system levels. Source: [Leitão and Restivo, 2008].

As supervisor holons provide coordination and optimisation services to other holons, they are able to introduce a specific hierarchy into a decentralised architecture. Thus, ADACOR enables an adaptive production control approach that is as centralised as possible and as decentralised as necessary. For that reason, two system states are introduced and

allow a hybrid control structure as given in figure 2.17. The stationary state is controlled by the coordination level and aims to achieve a global optimisation of the production process. The transient state, triggered after disturbances, resembles a heterarchical control architecture in terms of agility and adaptability [Leitão and Restivo, 2008]. ADACOR, in total, focuses on the prediction of disturbances and their intelligent handling.

ADACOR holons have an embedded self-organisation mechanism that includes local and global behaviours. Dynamic parameters locally reflect the degree of autonomy, and the learning capabilities allow a dynamic evolution of the local holon behaviour. The global self-organisation of the system is achieved by interaction of the individual holons using mechanisms inspired by ant behaviour, called stigmergy (cf. section 2.2.2) [Leitão and Restivo, 2008]. Besides this indirect coordination mechanism, explicit negotiation in form of negotiation protocols such as the Contract-Net Protocol (CNP) [Smith, 1980] is applied as well (cf. section 2.3) [Leitão and Restivo, 2008].

Going beyond the results published for ADACOR architecture in [Leitão et al., 2006, Leitão and Restivo, 2008], a holonic disturbance management architecture with a predictive dimension is introduced in [Leitao, 2011]. This dimension improves ADACOR concerning the re-scheduling after disturbance handling, and thus enhances reconfigurability and adaptability of the manufacturing system.

PROSA [Van Brussel et al., 1998] and ADACOR [Leitão et al., 2006], as holonic architectures, combine high predictability of hierarchies with the flexibility and reconfigurability of heterarchical systems by dynamically exploiting characteristics of both structures.

Concerning the MobComm objectives, ADACOR offers the same relevance as presented for PROSA [Van Brussel et al., 1998]. In addition to that, ADACOR emphasises even more the hybrid aspect by integrating different system states as a basis to implement maintenance of productivity (Objective 1). To show the limitations of ADACOR towards the MobComm objectives, figure 2.19 presents a combination between ADACOR and MobComm. The holonic manufacturing that is controlling a complete system can integrate a manufacturing component that is able to be functionally reconfigured. Focusing on an ADACOR Physical Holon [Leitão and Restivo, 2008], the MobComm commissioning robot can be used as a physical device and contributes to the total flexibility with a supplementary functional reconfigurability in the bottommost layer of the manufacturing system.

The selection of PROSA [Van Brussel et al., 1998] and ADACOR [Leitão, 2004] as

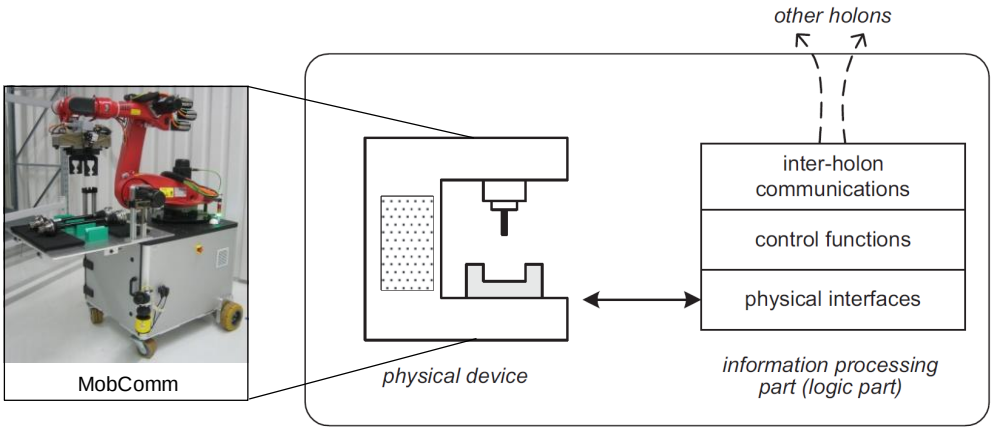


Figure 2.19: Possible combination of ADACOR physical holon and MobComm. Adapted from: [Leitão, 2004].

holonic manufacturing architectures finalises the presentation of manufacturing paradigms. After having started with the flexible and reconfigurable paradigms, the agent-based manufacturing including HMS was the focus of this review. The following section summarises and discusses the presented approaches with regard to their relevance for this work.

Discussion

A set of manufacturing paradigms was investigated in literature of the last few decades. Paradigms related to flexibility and reconfigurability have been presented in the previous section. Especially the reconfigurable manufacturing that further contains agent-based approaches and holonic manufacturing, has been focused on and been provided with a range of examples. The fulfilment of the research objectives and the characteristics of the individual approaches are summarised in table 2.2.

The strength of the listed agent-based and holonic approaches is their solution for a hardware-abstracted reconfiguration mechanism, as a part of Objective 3. By the application of an agent-based resource layer, the use of hardware functionalities is abstracted from the actually provided manufacturing components. Even if the mobile robot in this work is a manufacturing component itself, the concept of a resource layer is adapted in order to fulfil this part of Objective 3. In contrast to the resource abstraction, the functional reconfigurability cannot be solved by any of the presented approaches or is not explicitly evaluated.

For the implementation of a dependable integration of new skills, required in Objective

	Self-organisation (Obj. 1)	Maintenance of productivity (Obj. 1)	Dependability (Obj. 2)	Functional changes (Obj. 3)	Hardware abstraction (Obj. 3)	Characteristics
CoBASA [Barata and Camarinha-Matos, 2003]	Yes	part.	n/a	n/a	Yes	- Coalitions allow the generation of completely new control structures.
Agent-based Commissioning [Staab et al., 2004]	No	n/a	n/a	No	Yes	- Similar field of application. - Distributed commissioning scenario (hardware and software).
PABADIS'PROMISE [Feng et al., 2007]	Yes	n/a	n/a	n/a	Yes	- Decision making is shifted from ERP to MES.
PROSA [Van Brussel et al., 1998]	Yes	part.	n/a	No	Yes	- Hybrid architecture. - Decoupling of structure and algorithms.
ADACOR [Leitao and Restivo, 2008]	Yes	part.	n/a	No	Yes	- Reconfiguration and manufacturing activities decoupled.

Key: No = Objective not fulfilled, part. = Objective partly fulfilled, Yes = Objective fulfilled, n/a = No statement about objective available.

Table 2.2: Fulfilment of research objectives and specific characteristics concerning the presented approaches in the area of manufacturing paradigms.

2, no information is provided by these approaches. Due to the consistent use of agent technology and holons, self-organisation is implemented in all approaches in different levels, even though it is only a key characteristic in ADACOR [Leitão, 2004]. The holonic and thus hybrid control structures of HMS, given in ADACOR [Leitão and Restivo, 2008] and PROSA [Van Brussel et al., 1998], are relevant for MobComm as they provide the basic concept for the desired maintenance of productivity (Objective 1). Further, the strong use of coalitions in CoBASA [Barata and Camarinha-Matos, 2003] provide the foundation for a flexible change of control structures that has to be adapted to MobComm issues. Even if the fulfilment is only partial due to the missing parallelism of the given mechanisms, these concepts provide an initial situation for the research objectives in this work.

Additionally to the concept regarding the maintenance of productivity provided by mobile robotic area in section 2.1.4, the approaches presented in this section support the fulfilment of the maintenance of productivity (Objective 1) and offer solutions regarding the hardware abstraction of the system (Objective 3). Even though self-organisation

has already been implemented in the presented approaches, only ADACOR [Leitão and Restivo, 2008] highlights this aspect as a key characteristic. For this reason, the following section will focus on self-organisation of manufacturing systems including the presentation of implementation examples.

2.2.2 Self-Organisation in Manufacturing Systems

Motivated by the objective of a self-organised reconfiguration mechanism with integrated maintenance of productivity (Objective 1), self-organisation in manufacturing is reviewed in this section. The study of self-organisation was initiated as a field of exploration in the 1950s with the work presented in [Grasse, 1959], containing studies of insect society behaviours. The work of [Grasse, 1959] is regarded as the origin of biology-inspired self-organisation in various literature such as [Serugendo et al., 2003, Lopatkin, 2008, Valckenears et al., 2001]. Nowadays, self-organisation can be found in many fields such as MAS, grids, networking, robots, or manufacturing control [Serugendo et al., 2003].

The precise definition of self-organisation is still reasoned in literature as stated in [Correia, 2006]. The definition used in the present thesis is taken from an early and basic work about self-organisation:

Definition 2.12 (Self-Organisation) *A system is self-organising if it is self-managing (the system adapts to its environment without outside control), structure adaptive (the system establishes and maintains a certain kind of structure (e. g. spatial, temporal), providing the system's primary functionality), and employs decentralised control (the system has no central point of failure) [Zadeh, 1963].*

Based on this definition, the software structures resulting from the application of self-organisation are as well of interest for MobComm, given in definition 2.13:

Definition 2.13 (Self-organising software architecture) *A self-organising software architecture is one in which components automatically configure their interaction in a way that is compatible with an overall architectural specification. The objective is to minimise the degree of explicit management necessary for construction and subsequent evolution whilst preserving the architectural properties implied by its specification [Georgiadis et al., 2002].*

The presented characteristics of self-organisation lead to a complex system behaviour that emerges from the interaction of single components that themselves are only able to execute simple behaviours. This emergent behaviour allows a high level of flexibility and adaptability but impedes as well robustness and dependability in a system [Serugendo et al., 2003].

Especially in manufacturing a high level of self-organisation has to be examined critically regarding the required level of productivity. For this purpose, the controlled self-organisation is investigated over the last few years (e.g. in [Cakar et al., 2007]). The contradiction aimed to be solved is the increased degree of freedom due to self-organisation versus the required level of availability and robustness [Cakar et al., 2007].

Equally to the enhancement of flexibility and reconfigurability in the already presented manufacturing paradigms in section 2.2.1, agent technology is applied to realise self-organisation. Thus, this section focuses on self-organisation implemented in MAS [Serugendo et al., 2003] that can be divided into five different categories: Direct interaction, stigmergy, reinforcement, cooperation, and generic architectures [Serugendo et al., 2003]. The following description of these types further demonstrates the broad range of mechanisms investigated in the self-organisation area.

Direct interactions: Direct interaction uses basic principles like broadcasting or localisation. The change of agent structures, such as topological placement of agents and agent communication lines, is in its focus [Serugendo et al., 2006].

Stigmergy: Indirect interactions, like stigmergy, use biologically-inspired concepts that are broadly discussed in various literature like [Hadeli et al., 2003, Kumar and Cohen, 2004]. For the application of stigmergy, an appropriate and complex self-organisational model has to be defined, and needs to be verified by experimentation. In contrast to direct interaction, stigmergy is driven by changes in the environment [Serugendo et al., 2006]. [Valckenears et al., 2001] even states that the application of stigmergy integrates the environment as a part of the solution. Further, [Valckenears et al., 2001] gives an application of stigmergy in manufacturing control, based on PROSA architecture [Van Brussel et al., 1998](cf. section 2.2.1). Stigmergy is implemented as an ant-based propagation mechanism [Valckenears et al., 2001], illustrated in figure 2.20. A mobile agent moves virtually through a manufacturing system and propagates the actual production order to the available resources.

Reinforcement: Reinforcement mechanisms dynamically modify the agent behaviour

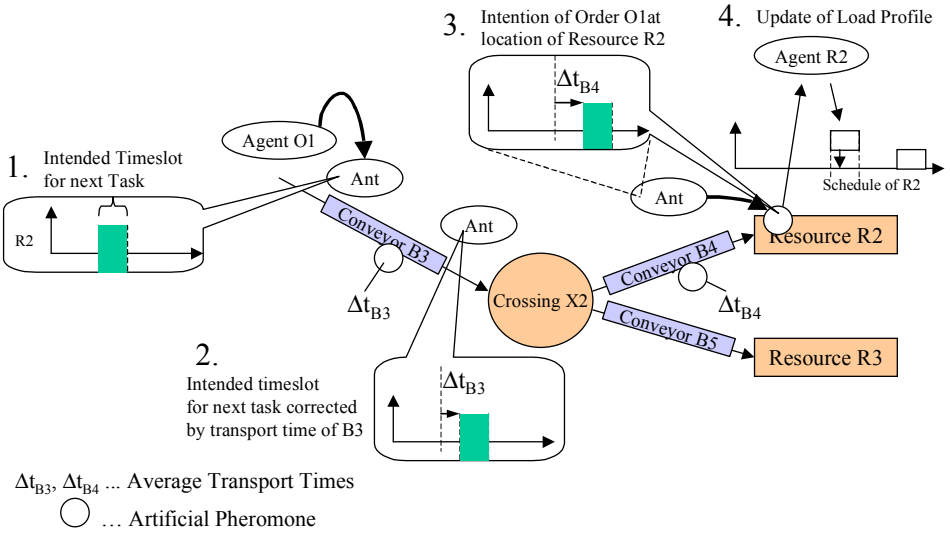


Figure 2.20: Application of stigmergy as an ant-based propagation mechanism. Source: [Valckenears et al., 2001].

according to arising reinforcement, which implies that rewards increase the specific agent behaviour and punishment decreases it. Individual agents can adapt their own capabilities depending on these external changes [Serugendo et al., 2006].

Cooperation: The cooperation mechanism can be simplified as dynamic compositions and decompositions of agents. A decomposition implies the division of an agent into two and can be performed to respond to overwhelming environmental demands. In contrast, the composition merges two agents into one and can be useful when the communication overhead between two agents is too high. In this mechanism, the single agent has no global view and the system has no central control [Serugendo et al., 2006].

Generic architecture: Using a generic architecture, meta-models of the agent organisations are instantiated and dynamically modified for specific applications. Architectures like ADACOR [Leitão and Restivo, 2008] or PROSA [Van Brussel et al., 1998], presented in section 2.2.1, are examples for this mechanism. Self-organisation hereby signifies the change of the holarchy caused by environmental changes such as hardware disturbances or hardware breakdowns [Serugendo et al., 2006].

For a detailed evaluation of the applied mechanisms going beyond this description, the reader is referred to [Serugendo et al., 2006].

Even if the application of self-organisation varies in domains and mechanisms, some generic prerequisites have to be generally fulfilled in engineered systems to implement self-organisation. These requirements, discussed in different literature [Serugendo et al.,

2006,De Wolf and Holvoet, 2005,Correia, 2006], are listed in the following, taken from [Frei et al., 2007c]:

- Autonomous and interacting units with no external control.
- Positive and negative feedback mechanisms for different interpretations.
- System variations that lead to far-from-equilibrium state like disturbances or changing production requirements.
- Mechanical and logical safety measures in case of unwanted or harmful behaviour.
- A flat internal architecture with dynamically changeable coalitions of agents. [Frei et al., 2007c].

After the listing of generic requirements to implement self-organisation, EAS [Frei et al., 2007c] and the Restore Invariant Approach [Guedemann et al., 2006] are presented as examples of highly developed self-organising manufacturing systems.

Evolvable Assembly Systems

Evolvable Assembly Systems (EAS) and a set of manufacturing paradigms have been introduced in section 2.2.1. Referring to figure 2.7, EAS are assigned to decentralised manufacturing similar to the agent-based or holonic principles.

EAS focus on assembly systems, a subgroup of manufacturing systems as given in definition 2.4 additionally to the terms evolvability and EAS.

Definition 2.14 (Assembly System) *An assembly system is an industrial installation able to receive parts and join them in a coherent way to form the final product. It consists of a set of [...] modules such as conveyors, pallets, simple robotic axes for translation and rotation as well as more sophisticated industrial robots, grippers, sensors of various types, etc. [Frei et al., 2008].*

Definition 2.15 (Evolvability) *Evolvability means the ability of complex systems to co-evolve with changing requirements, to undergo modifications of different significance, from small adaptations on-the-fly to more important transformations [Frei et al., 2007a].*

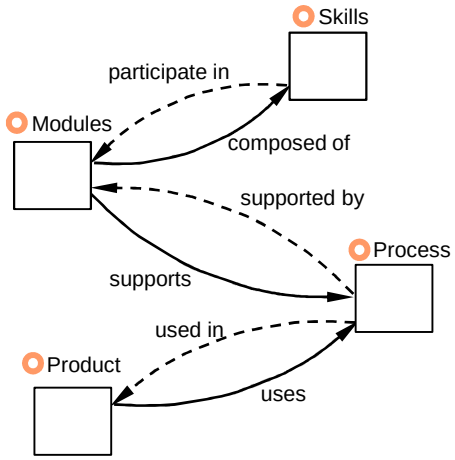
Definition 2.16 (Evolvable Assembly System) *An EAS is an assembly system that can co-evolve together with the product and the assembly process. It can easily undergo*

small and big changes in product design and seamlessly integrates new modules independently from their brand or model [Frei et al., 2008].

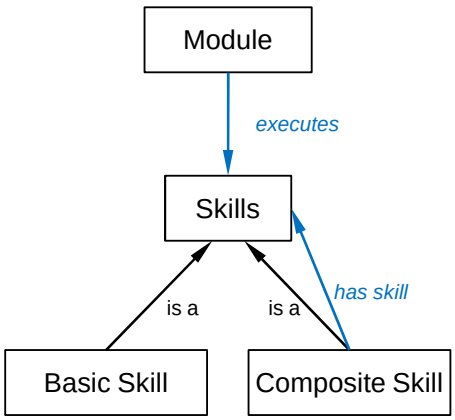
The main goal of evolvable assembling is the dynamic allocation and sequencing of tasks, and the dynamic load adaptation [Barata et al., 2006]. Therefore, it contains a reconfigurable system platform that exhibits emergent behaviour and is able to automatically determine the functionality of the system by evaluating the skills of the component [Onori et al., 2006]. The following passage shows how evolvability combined with self-organisation is used to accomplish the named EAS goals.

EAS parts consist of products, processes, manufacturing components, called modules, and skills. Figure 2.21(a) shows these parts and their relations. The loosely coupled parts have no predefined system structure, instead, the resulting assembly system is formed by a set of process-oriented modules due to given requirements [Onori et al., 2006].

Every module is composed of a set of skills that is used to map the specific functionalities, as illustrated in figure 2.21(a). By considering the internal skill structure, pictured in figure 2.21(b), EAS are able to compose complex assembly skills by the re-use of simpler ones. For example, the failure or withdrawal of modules can lead to a missing skill. To compensate the absence of a module, the exiting skills collaborate dynamically, and form coalitions to be able to offer the according functionality as a composite skill to the system.



(a) Overview of EAS components. Adapted from: [Barata et al., 2006]



(b) Skill structure in EAS. Adapted from: [Semere et al., 2007]

Figure 2.21: Overview and skill structure in EAS.

Compared to traditional EAS, self-organising EAS (SO-EAS) exhibit two additional characteristics. The modules self-organise the appropriate assembly layout, and the system

self-manages the execution of the assembly tasks by the adaptation to changing production conditions [Frei, 2010]. To achieve these characteristics, self-organisation of classical EAS is enhanced with metadata and policies. Metadata map functionalities and performance of the modules additionally to data owned by the modules. Thus, metadata contains self-descriptions of skills, interfaces, and performance levels of skill coalitions. The policies, however, underlay a dynamic enforcement at production time, and build the basis for the decision-making and adaptation process in SO-EAS. Agentified modules, metadata, and policies are decoupled from each other to be dynamically changeable as presented in figure 2.22. For the purposes of evolution, SO-EAS cannot always guarantee a constantly stable state [Frei et al., 2007b].

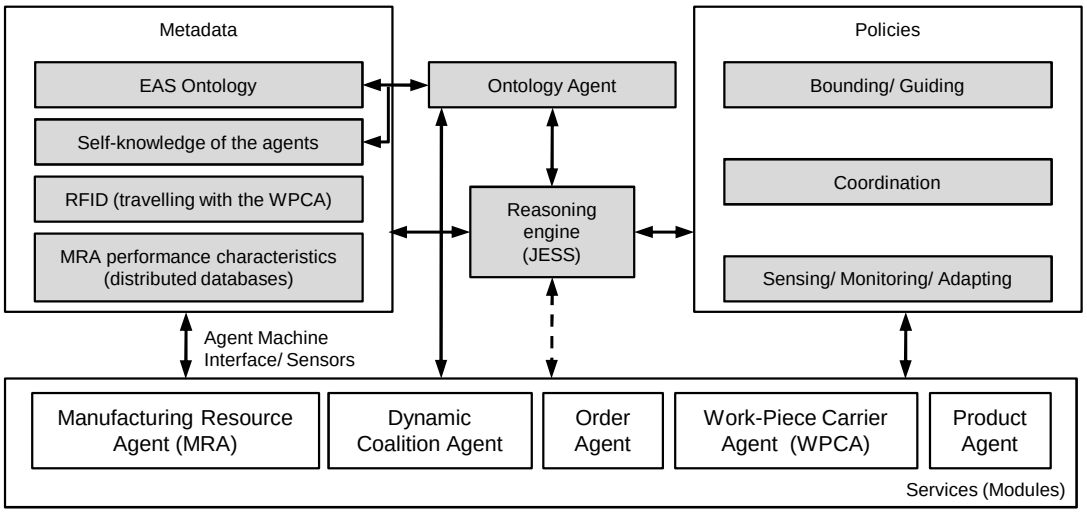


Figure 2.22: Architecture of a SO-EAS. Adapted from: [Frei et al., 2009].

The evolvable manufacturing paradigm is applied and further developed in the EU-project EUPASS that aims at creating a framework for rapid integration of ultra-precision assembly modules. Results of EUPASS are summarised in the project report [EUPASS, 2008] and publications such as [Wehrli et al., 2008]. As the field of application, i.e. the ultra-precision assembly modules, differ considerably from MobComm, further details are renounced in this review.

The relevance of SO-EAS for MobComm is the provision of a reconfigurable system platform and the self-organised coalitions of functional skills for the generation of more complex functionalities that are a possible basis for the self-organised reconfigurability in MobComm (Objective 1). Beyond that, the reconfigurable platform allows hardware-abstraction (Objective 3). Due to the high degree of evolution, a SO-EAS cannot guarantee

a constantly stable state which limits this approach (Objective 2). The second limitation is the focus on assembly systems that provide different requirements than the manufacturing component level in MobComm. Even if functional skills are provided, no information about the reconfiguration after functional process changes is provided in literature (Objective 3).

The second application presented in this section is the Restore Invariant Approach (RIA) that is related to EAS regarding the enhancement of flexibility and the self-managing properties in manufacturing.

Restore Invariant Approach

The research area of Organic Computing (OC) tries to eliminate the limitation of EAS, which is the missing predictability of system states. RIA is based on OC, and has developed the vision of a robust and dynamic adaptation of self-organising systems to changing environments without running out of control. The realisation of this vision allows to overcome the described limitation. Organic Computing is defined as follows:

Definition 2.17 (Organic Computing) *An organic computer is a self-organised system that can adapt to a dynamically changing context and achieves the self x-properties of Autonomic Computing [Sterritt, 2005]: self-configuration, self-optimisation, self-healing, self-explanation, and self-protection [Richter et al., 2006].*

Due to the focus on RIA that is based on OC, Autonomic Computing is not further detailed and the interested reader is referred to literature such as [Sterritt, 2005, Kephart and Chess, 2003]. In summary, the most significant difference between Autonomic and Organic Computing is the ability of OC to react sensibly to external requirements and to keep the behaviour of the system within certain boundaries by taking the effects of emergent behaviour into account. This effect is not considered in previously presented approaches like EAS [Barata et al., 2006] or ADACOR [Leitão and Restivo, 2008].

The internal system organisation of OC is the observer/controller architecture. This architecture gives a regulatory feedback to control the dynamics of the system in case of emergent and thus uncontrollable behaviours. The basic decentralised system in such an architecture is a System under Observation/Control (SuOC), as given in figure 2.23.

Observer and controller are responsible for the appropriate surveillance and feedback of the system and allow the implementation of a controlled self-organising system [Richter et al., 2006], as introduced at the beginning of this section. As stated in [Serugendo

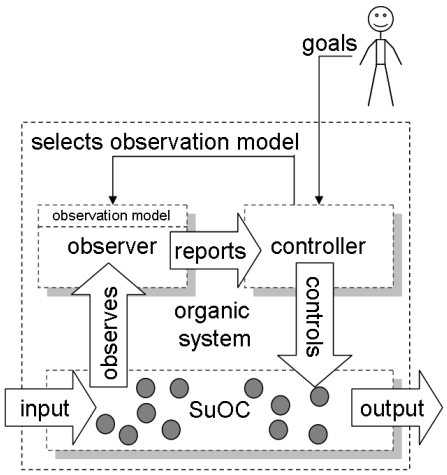


Figure 2.23: Overview of an observer/controller architecture. Source: [Richter et al., 2006].

et al., 2008], the idea of an observer/controller architecture is also implemented in a SO-EAS [Frei et al., 2007c]. The enforcement of SO-EAS policies acts as the controller and the monitoring of metadata is the observer in the evolvable paradigm.

The RIA presented in this section is inevitably connected to the Organic Design Pattern (ODP) that constitutes a design and construction guideline for self-managing systems, the RIA is based on. ODP are applied to RIA and thus explained prior to the presentation of the invariants restoration.

ODP deal with agents that process workpieces, called resources, with one or more of the agents' capabilities according to a given task. A task, however, describes the way a given workpiece has to be processed. The according task description is a sequence of capabilities needed for the treatment of the workpiece. Every agent is defined by its capabilities and by the agents it can receive from or deliver to its workpieces. The capability an agent performs in a specific situation is determined by its role. The *RoleAllocation*, important for the compliance of the total manufacturing goal, is the complete set of agent roles [Guedemann et al., 2008], as given in the diagram of figure 2.24.

While applying ODP, the reconfiguration of a manufacturing system is conducted with the RIA. The recognition of a needed reconfiguration is based on the violation of a logical formula - the invariant. An invariant contains the actual configuration and capabilities of the set of agents, and carries the intended tasks as free variables. The invariants are hold true as long as possible by the allocation of the set of free variables [Guedemann et al., 2008]. In case an invariant is evaluated as false, e.g. caused by a disturbance or hardware breakdown, an event is triggered to restore it. The observer/controller layer, responsible

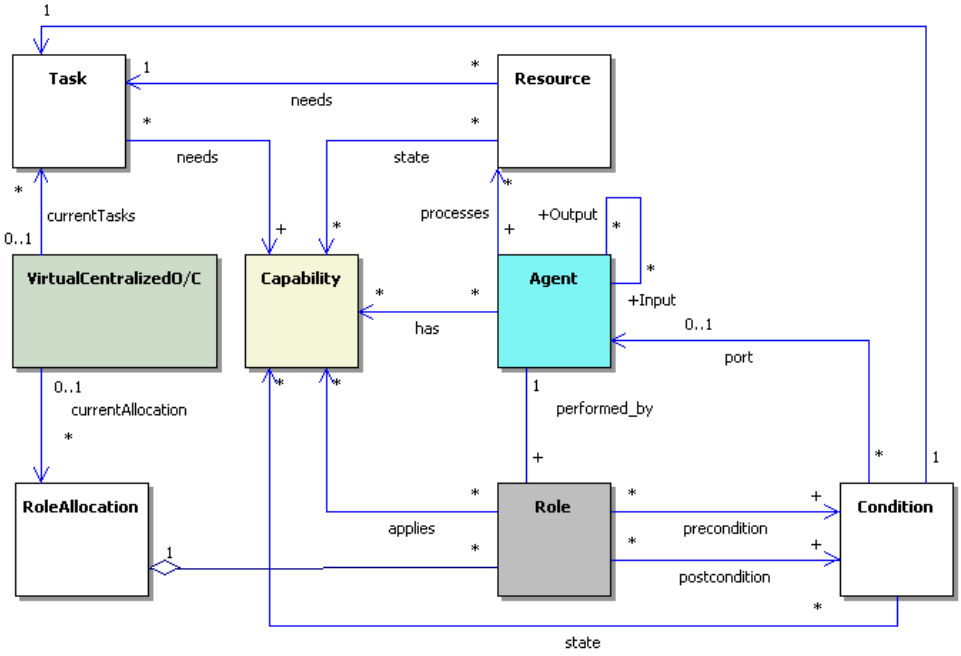


Figure 2.24: Object model of the ODP. Source: [Guedemann et al., 2008].

for calculating the new *RoleAllocation*, is required to get the system from reconfiguration to operation again. As every invariant consists of static and dynamic variables, the O/C layer analyses the allocation of the free variables with respect to the changes that caused the reconfiguration, and restores the invariant accordingly [Nafz et al., 2009]. As introduced in the literature review in section 2.1.3, RIA makes use of the domain-independent Linear Temporal Logic for the specification of the invariants and their restoration.

The RIA also provides a self-adaptation to new tasks, relevant for the desired functional reconfiguration in MobComm (Objective 1) but is limited to dynamic changes in sequence and iteration of the task [Guedemann et al., 2008]. No adaptation to functional process changes can be accomplished. Figure 2.25 gives a possible combination of RIA and MobComm. Besides the already provided reconfiguration possibilities in RIA, this combination will be able to handle functional process changes as presented in the example task change in figure 2.25. As a result, RIA will provide the capability to handle functional, task sequence, and task iteration changes additional to hardware breakdowns and disturbances.

As the observer/controller layer does not re-initialise the operational state until a new configuration has been located [Nafz et al., 2009], an undesired loss of productivity, contrary to the desired maintenance of productivity (Objective 1), has to be expected.

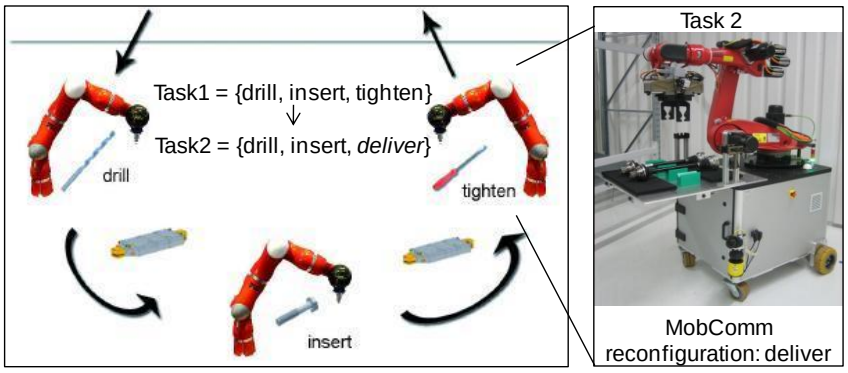


Figure 2.25: Combination of RIA and MobComm. Adapted from: [Guedemann et al., 2008].

Besides this limitation, the dependable handling of reconfiguration events by using an observer/controller architecture is of importance for the dependable integration of new skills in MobComm (Objective 2).

After the presentation of EAS [Barata et al., 2006] and RIA [Guedemann et al., 2006] as selected self-organising manufacturing systems, their relevance for the present thesis is further discussed in the following section.

Discussion

For the review of self-organisation in manufacturing systems, two approaches have been selected. EAS [Barata et al., 2006] and RIA [Guedemann et al., 2008] are summarised in table 2.3 including their characteristics and levels of objective compliance.

	Self-organisation (Obj. 1)	Maintenance of productivity (Obj. 1)	Dependability (Obj. 2)	Functional changes (Obj. 3)	Hardware independence (Obj. 3)	Characteristics
EAS [Barata et al., 2006]	Yes	part.	No	n/a	Yes	- Recombination of basic skills in assembly systems to create new composite skills.
RIA [Guedemann et al., 2008]	Yes	No	Yes	No	Yes	- Use of invariants to dependably control the behavior of the system.

Key: No = Objective not fulfilled, part. = Objective partly fulfilled, Yes = Objective fulfilled, n/a = No statement about objective available.

Table 2.3: The summary of self-organisation in manufacturing systems.

According to their key focus, both approaches are able to comply with the self-organisational requirements of Objective 1. The reconfigurable platform used in EAS [Barata et al., 2006] is not a satisfying solution for the desired maintenance of productivity but provides a basis for further investigation and adaptation. Similar to the agent-based and holonic approaches, discussed in section 2.1.4, hardware abstraction is a given functionality of both implementations in contrast to the functional reconfigurability that is not evaluated or not provided (Objective 3). The high level of dependability applied in RIA [Guedemann et al., 2006] provides a comprehensive solution for Objective 2.

Including the positive evaluations of self-organisation (Objective 1), hardware-abstraction (Objective 3) and dependability (Objective 2), the majority of requirements can be partially or completely fulfilled by the self-organising manufacturing area. This emphasises the relevance of this reviewed area for MobComm.

After presenting established manufacturing paradigms with a focus on reconfigurability and flexibility in section 2.2.1, and controlled self-organisation in this section, the knowledge engineering aspect of manufacturing systems is reviewed in the following.

2.2.3 Knowledge Engineering in Manufacturing Systems

Reconfigurability after functional process changes and the resulting integration of new functionalities into the system implicate the integration of semantics and knowledge which motivates the review of the following area.

Starting with general aspects of knowledge engineering, the CommonKADS approach [Post et al., 1997] provides a standardised methodology for structured knowledge engineering. It presents the challenge of knowledge representation as its appropriate modelling that is less a large and flat database but rather a fine-grained knowledge base divided into similar-structured partitions [Schreiber et al., 1999]. The applied knowledge structure is the basis for knowledge analysis, validation, and maintenance. The desired similar-structured knowledge base can be achieved by either data or ontology models [Schreiber et al., 1999]. Compared to data models that are not intended to be shared by other applications, ontology models are generic and task-independent [Spyns et al., 2002] that allows their use in application-crossing reconfigurations.

The following definition of an ontology, provided by [Gruber, 1993], is well-established in literature:

Definition 2.18 (Ontology) *An ontology is a formal specification of conceptualisation [Gruber, 1993].*

The term ontology is borrowed from philosophy where an ontology is a systematic account of existence. Adapted to knowledge-based systems it means that "what *exists* can be represented as well" [Gruber, 1993]. Ontologies are able to identify and formalise generic components that can be reused across different domains to support a robust development of knowledge based systems [Motta and Lu, 2000]. Beyond that, they facilitate verification, validation, and reuse of knowledge in new systems [Bench-Capon, 1998]. For further reading in the steadily growing research field of ontologies, the user is referred to [Uschold and Grüninger, 1996, Evermann and Fang, 2010].

The motivation of ontology use in manufacturing systems does not differ from their general provision of shared knowledge, and the ease of application knowledge integration [Xuemei, 2007]. Especially the agent-based manufacturing is dominated by ontologies, in contrast to mobile robots that sparsely use ontologies as their internal knowledge representation [Parker, 2008]. Physical robots are more challenged by uncertainties, limited power, or computation problems than by unsolved shared knowledge. A prerequisite for the use of ontologies in industrial mobile robots like the MobComm robot, is a conceptualisation of the system capabilities [Parker, 2008].

SIARAS [Bengel, 2007], Plug and Produce [Naumann et al., 2007], and the ontology-based reconfiguration agent [Alsafi and Vyatkin, 2010] are presented in the following as examples of ontology-driven systems in manufacturing.

SIARAS

The skill-based inspection and assembly of reconfigurable automation systems (SIARAS) [Bengel, 2007, Bengel, 2009] present the work of the European SIARAS project that develops a SkillServer to support the expert-controlled reconfiguration of existing manufacturing systems. Using ontologies, SIARAS reasons about modified manufacturing requirements in a given process. SIARAS [Bengel, 2007] follows the idea of skill-based manufacturing where manufacturing components have embedded knowledge about their self-aware skills and interact to solve a task together.

The goal of SIARAS is the suggestion of new system configurations to the user after changed tasks. By querying the ontology, a changed task description is translated into a

definition, understandable by the *SkillServer* [Malec et al., 2007] that reviews if the task description can be satisfied by the current set of operations or if a reconfiguration has to be initiated. *SkillServer* and ontology, as the core components of SIARAS, are pictured with the complete SIARAS architecture in figure 2.26.

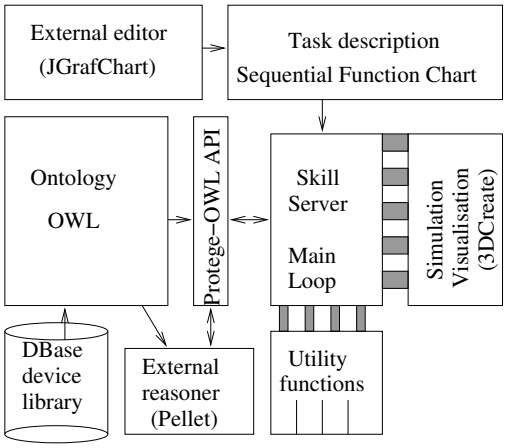


Figure 2.26: Overview of SIARAS components. Source: [Malec et al., 2007].

The first step of the SIARAS reconfiguration results in the suggestion of a set of skills for the changed task description. For each operation on the workpiece the user has to manually select one of the suggested skills whereas this selection is embedded in a CAD (Computer Aided Design) application for enhanced usability. After this selection, the second part of the reconfiguration is initiated by searching appropriate manufacturing components for the requested skills. Possible parameters, properties, and restrictions of the manufacturing components are evaluated by querying the ontology. Before the reconfiguration is finalised, the set of found components is filtered by quality factors implemented in the utility functions of the system [Bengel, 2009].

The limitation of SIARAS regarding MobComm is the high impact of expert knowledge during the reconfiguration process, set in contrast to the required self-organisation (Objective 1). Despite this limitation, SIARAS offers a skill-based reconfiguration that allows an ontology-based encapsulation of system functionalities, and can serve as a basis for a functional reconfiguration in this work (Objective 3). Beyond that, SIARAS is relevant for this thesis as the *SkillServer* is able to conduct a validity check to raise dependability (Objective 2) of the newly created operations [Malec et al., 2007].

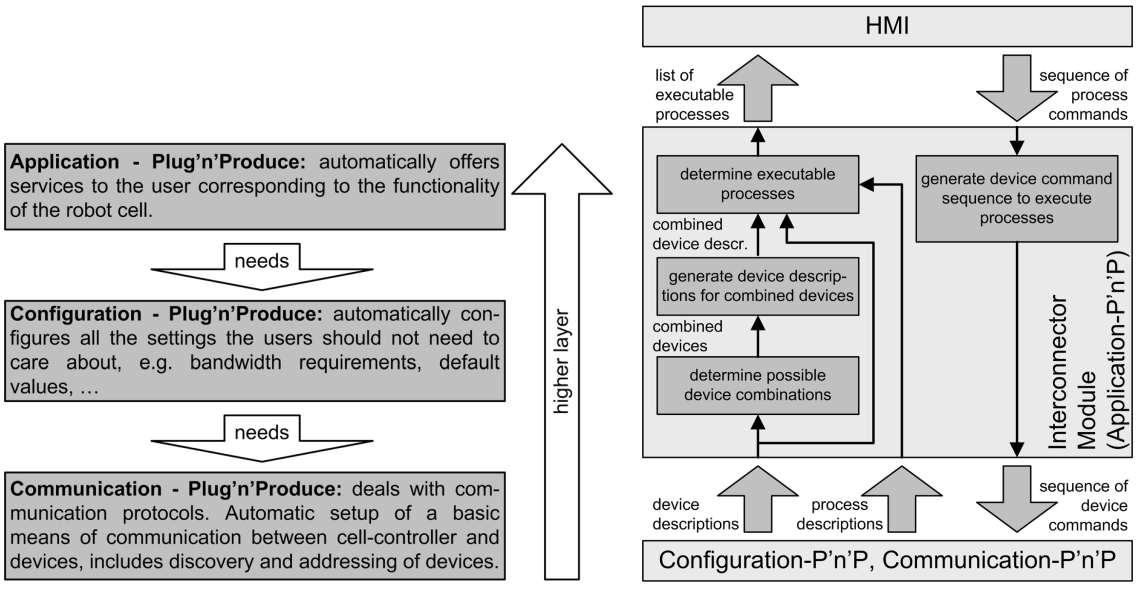
In addition to SIARAS, the Plug and Produce approach [Naumann et al., 2007], also aiming a simplified reprogramming of existing manufacturing systems, is presented in the

following.

Plug and Produce

The Plug and Produce (P'n'P) approach offers easy means of reprogramming to users of a robot cell in small and medium enterprises (SME) without the need of communication or configuration knowledge. The generic P'n'P layers, as given in figure 2.27(a), enable this user-friendly approach. Application, communication, and configuration layers are separated. This section focuses on the connection between user and application in the Application-P'n'P layer [Naumann et al., 2007]. P'n'P concentrates on fast adaptability of robot cells to changed task descriptions inserted by the user as a sequence of process commands. These commands are transformed into a list of executable processes, illustrated in figure 2.27(b), that can finally be offered to the user for manual selection.

The core part of P'n'P is the Interconnector that handles the user descriptions as inputs, evaluates them automatically, and generates the set of executable processes as an outcome [Naumann et al., 2007]. Similar to SIARAS [Bengel, 2007], P'n'P requires a high degree of user interaction compared to approaches like RIA [Guedemann et al., 2008] or EAS [Barata et al., 2006] that include self-organisation.



(a) Generic P'n'P layers.

(b) P'n'P modules and relations.

Figure 2.27: Overview of P'n'P layers and modules. Source: [Naumann et al., 2007].

The conversion of user commands into executable processes is mainly based on state-chart models and has to be supplied as a component description for every manufacturing

component including the assigned skills. By using state charts possible component combinations and a feasible set of executable processes can be extracted from the system. For a new combination of components a unique component description has to be produced by using the single state-charts of the used manufacturing components. A new description resulting of single state-charts can be generated since the process and component state-charts are linked via ontology. This allows to identify their interdependencies and synchronisation needs, also in new combinations [Naumann et al., 2007].

Even if P'n'P methodology differs from SIARAS [Bengel, 2007], the ontology-based reconfigurability is similar in goals and outcome. A unique characteristic of P'n'P with relevance to MobComm is the recombination of different components using the dynamic mapping of synchronised state-chart diagrams. This methodology facilitates a high degree of dependability as desired in Objective 2.

The ontology-based reconfiguration agent [Alsafi and Vyatkin, 2010] is presented as a third example and finalises the presentation of ontology-based manufacturing in the next section.

Ontology-based Reconfiguration Agent

The ontology-based reconfiguration agent is an intelligent reasoning software agent that allows adapting to changes in the manufacturing requirements or environment. The provided agent architecture enables the integration of high-level planning into distributed low level control which generates an alternative sequence of operations or a new feasible system configuration after changes [Alsafi and Vyatkin, 2010]. The interaction with the ontological model of the environment, with the floor specification, and with the manufacturing requirements is the basis of this approach. The named sources are presented in the bottommost agent layer in figure 2.28.

Decision, analysing and modelling, and specification layers are needed to decide whether an operation can be supported by the manufacturing environment or not. All layers depend on the layer beneath and use their capabilities and features. The specification layer provides all knowledge about the process and the environment that is transformed into an implicit formatted information by the analysis and modelling layer. Thereof a concrete list of available machines is the output of this middle layer. The topmost decision layer is responsible for intelligent reasoning and thus for the generation of the new configuration. This layer further divides the generated configuration into sub-configurations according to

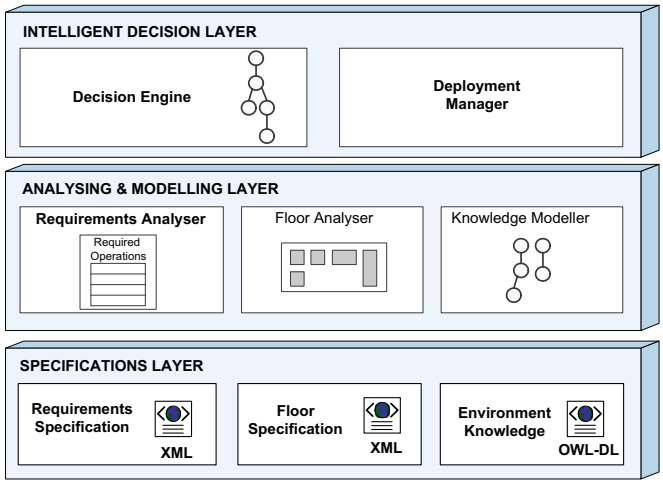


Figure 2.28: Layers of the ontology-based reconfiguration agent. Source: [Alsafi and Vyatkin, 2010].

the distribution of the controllers [Alsafi and Vyatkin, 2010].

Even if the approach given in [Alsafi and Vyatkin, 2010] is limited regarding the missing functional reconfiguration (Objective 3), it allows the flexible integration of input variables like manufacturing and environment descriptions to enhance reconfiguration flexibility, relevant for MobComm.

SIARAS [Bengel, 2007], P'n'P [Naumann et al., 2007], and the ontology-based reconfiguration agent [Alsafi and Vyatkin, 2010] have been presented as selected examples for knowledge engineering in manufacturing and will be further discussed in the next section regarding their relevance for MobComm.

Discussion

As a third aspect of manufacturing systems research, knowledge engineering was reviewed in the previous section. SIARAS [Bengel, 2007], P'n'P [Naumann et al., 2007], and an ontology-based reconfiguration agent are summarised in table 2.4 including their level of objective compliance.

The main relevance of the knowledge engineering aspect for MobComm is the provision of dependability and thus the fulfilment of Objective 2. SIARAS [Bengel, 2007] uses a SkillServer enhanced with utility functions and P'n'P [Naumann et al., 2007] reaches dependability by a formal synchronisation of state-chart diagrams. The high relevance towards the MobComm dependability is in contrast to the missing self-organisation and the lack of concepts to maintain productivity in a reconfigurable system (Objective 1). Focused

	Self-organisation (Obj. 1)	Maintenance of productivity (Obj. 1)	Dependability (Obj. 2)	Functional changes (Obj. 3)	Hardware independence (Obj. 3)	Characteristics
SIARAS [Bengel, 2007]	No	No	Yes	part.	Yes	- Generation of new/adapted skills due to known functionalities. -High degree of expert knowledge required.
Plug and Produce [Naumann et al., 2007]	No	No	Yes	n/a	Yes	- Generation of new executable processes by combining given processes.
Ontology-based Reconfiguration Agent [Alsafi and Vyatkin, 2010]	Yes	n/a	part.	No	Yes	- Knowledge-based reconfiguration after changes in floor specification, requirement specification, and environment specification.

Key: No = Objective not fulfilled, part. = Objective partly fulfilled, Yes = Objective fulfilled, n/a = No statement about objective available.

Table 2.4: The summary of knowledge engineering in manufacturing systems.

on a reconfiguration after hardware changes and changed sequences, only SIARAS [Bengel, 2007] with its ontology-based encapsulation of system functionalities is able to provide a solution idea for the desired functional reconfiguration in MobComm (Objective 3). The remaining approaches do not state or not fulfil functional reconfigurability.

Even if the presented ontology-based mechanisms can not comply with a set of MobComm objectives like self-organisation or the maintenance of productivity (Objective 1), concepts provided for a high dependability state the relevance of this research aspect. They complement the solutions provided by RIA [Guedemann et al., 2006] regarding a dependable reconfigurability (Objective 2).

The discussion of knowledge engineering in manufacturing finalised the whole manufacturing systems review. The different manufacturing paradigms presented in section 2.2.1 contribute to the investigation of the maintenance of productivity (Objective 1) and hardware abstraction (Objective 3), the self-organisation aspect in section 2.2.2 has its strength in the self-organised mechanism as expected (Objective 1), completed by the knowledge engineering aspect in this section implementing high dependability (Objective 2).

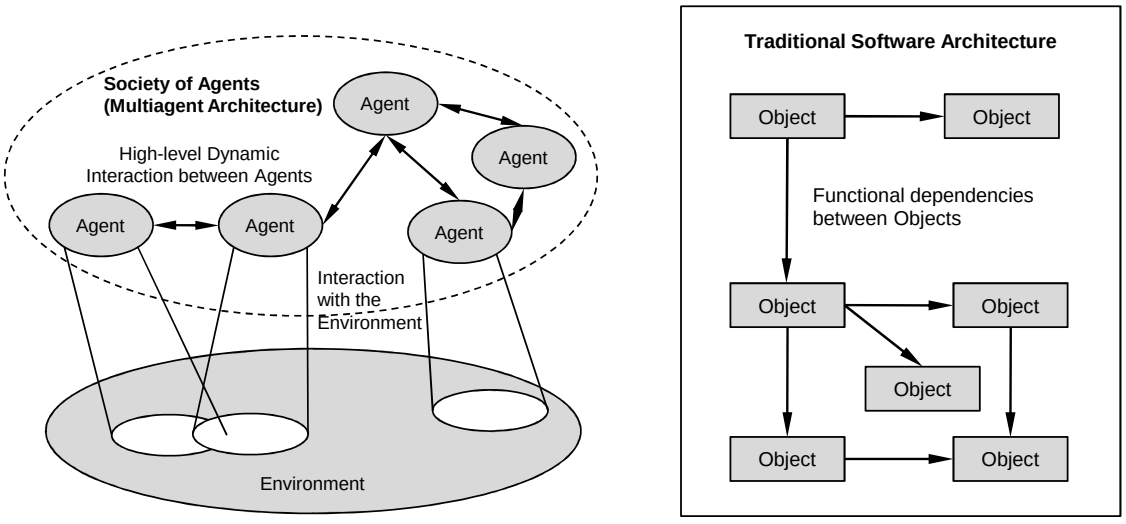
The next section about agent-oriented software engineering is motivated by the consistent use of agent technology in the relevant approaches for this thesis.

2.3 Agent-Oriented Software Engineering

The third and last research area within related work is the Agent-Oriented Software Engineering (AOSE). This review is less focused on the set of research objectives but rather on the implementation of the resulting system, mapping the set of research objectives to a system implementation, that can be evaluated. The implementation of MobComm is a central research aim and requires the use of appropriate software engineering techniques.

Following the approaches presented in section 2.1 and section 2.2, the use of agent technology is emphasised consequently to realise self-organisation, flexibility, and reconfigurability such as ADACOR [Leitão and Restivo, 2008] or RIA [Guedemann et al., 2006]. This widespread use of agents and their proven applicability is the motivation for their use in MobComm and the need of AOSE in this review. After a general introduction to the topic, already presented approaches are analysed regarding their used software engineering techniques. Following the results of this analysis, agent platforms, agent interaction, and FIPA standards are presented. The review is finalised with the discussion in section 2.3.3.

A comparison between traditional and agent-based software systems in figure 2.29 presents the main differences between agent-oriented and classical software development techniques. Further, it is highlighted why most of the object-oriented techniques mismatch with the application of agent technology as stated in [Wooldridge et al., 2000].



(a) Agent-based software system.

(b) Traditional software system.

Figure 2.29: Comparison between agent-based and traditional software systems. Adapted from: [Zambonelli and Omicini, 2004].

The agent-based architecture contains agents as defined in definition 2.9 that are situated in an environment. They can flexibly achieve their goals by interacting with others in terms of protocols and languages. A traditional software approach, however, as given in figure 2.29(b), shows functional dependencies between the used objects. Today's distributed and concurrent software systems cannot be strictly classified as traditional approaches any more as the used objects and components are more viewed as agents than as traditional objects. Especially the aspect-oriented programming paradigm overcomes the functional decomposition of traditional software systems. Through context-dependencies in component-based applications, the distinction between agents and their environment is nearly achieved [Zambonelli and Omicini, 2004]. Thus today, complex object-oriented systems appear more like a dynamic society than a static software architecture, and resemble to a great extend the agent-oriented paradigm. Independent of the distinctive level of autonomy and interaction, software systems that follow the architecture given in figure 2.29(a), apply different engineering techniques to achieve their goals.

The research area of agent technology goes back to the 1990s where publications like [Wooldridge and Jennings, 1995] and [Ferber, 1999] determined the advancements in this area. The engineering process of a MAS is mostly focused on the overall behaviour, whereas the single agent behaviour is less interesting [Zambonelli and Omicini, 2004]. Nevertheless, the reliance on a controllable and predictable behaviour of single agents and their interactions is the basis for a successful engineering process.

The presented agent techniques in this section are deduced from the implementation analyses of the already presented approaches in section 2.2. The used agent platforms and tools of the approaches, identified as relevant for MobComm, are summarised in table 2.5.

An agent platform builds the basis for any MAS application while tools like reasoning engines (e.g. Jess, Jena) or ontology editors (e.g. Protégé) facilitate the implementation of additional system features. Regarding the used agent platforms in table 2.5, the Java Agent Development Environment (JADE) [Bellifemine et al., 2000] can be extracted as a de-facto standard in literature, as confirmed in [Bordini et al., 2006] or [Vallejo et al., 2010]. The majority of the relevant approaches use JADE including Jadex platform, a JADE extension. This widespread agent platform and its extension Jadex are further investigated in the following section.

Related work	Implementation	Agent platform	Additional tools
CoBASA [Barata and Camarinha-Matos, 2003]	Yes	Jade	Jess
Agent-based commissioning [Staab et al., 2004]	Yes	Jini	
PABADIS [Feng et al., 2007]	No		
PROSA [Van Brussel et al., 1998]	No		
ADACOR [Leitao and Restivo, 2008]	Yes	Jade, Jadex	Jess
EAS [Barata et al., 2006]	Yes	Jade	Jess/Jena, Protégé
ODP/Restore invariant approach [Guedemann et al., 2008]	Yes	Jadex	
Ontology-based Reconfiguration Agent [Alsafi and Vyatkin, 2010]	Yes	Not described	Jdom, Pellet, Jena, Protégé

Table 2.5: Summary of agent platforms and tools used in MobComm related approaches.

2.3.1 Agent Platforms

Even if this section only presents JADE and Jadex, a large number of other agent platforms are available in literature and cover different requirements. The most influencing platforms, summarised in [Vallejo et al., 2010], are Cougaar [Helsinger and Wright, 2005], Agent Factory [Collier, 2002], and JACK [Winikoff et al., 2002] in addition to JADE [Bellifemine et al., 2007] and Jadex [Pokahr et al., 2005]. Characteristics of this variety of platforms are detailed in [Vallejo et al., 2010] and not further focus of this review.

JADE

JADE is a Java-based agent platform that was originally specified to implement the standards of the Foundation of Intelligent Physical Agents (FIPA). Since the late 1990s, a large number of applications in different domains is implemented with JADE that provides a GUI-based remote platform management for convenient administration. According to FIPA standards, the FIPA Agent Communication Language (ACL) is used for message communication. FIPA standards and their interaction protocols are covered in section 2.3.2.

The main components of JADE are the agent management system (AMS), the directory facilitator (DF), and the communication channel. JADE agents themselves are implemented as threads and hosted in agent containers that provide the runtime environment for them. The classical JADE agents are behaviour-based which implies that they

are not able to change or adapt their assigned behaviours in runtime. However, agent mobility including code and agent states is supported [Bellifemine et al., 2007]. Due to the full exploitation of Java and the provision of an object-oriented API, the engineering of JADE-based implementations does not require AOSE specific methodologies [Nowostawski et al., 2000]. Figure 2.30 gives the overview of JADE layers and components including the set of JADE containers.

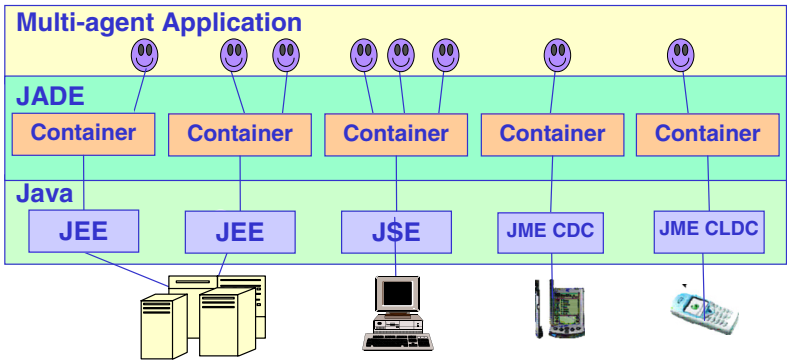


Figure 2.30: Overview of JADE components and layers. Source: [Bellifemine et al., 2008].

This homogeneous JADE layer hides complexity and diversity of the underlying Java-layer from the agent applications in the topmost MAS-layer [Bellifemine et al., 2008].

As JADE is the most widespread agent platform, a large number of implementations (exceeding those given in table 2.5) are given in literature. To underline the universality of JADE as its main strength, the implementation of a generic negotiation agent is chosen as an example in the following.

The presented negotiation agent derives its benefit from dynamic negotiation protocols and negotiation strategies that allow to flexibly adapt in a system to changing requirements [Paprzycki et al., 2004]. The resulting dynamic loading of the different reasoning models into the negotiation agent is presented in figure 2.31.

Communication is the only static module and contains the standardised FIPA ACL. The changeable protocol modules are composed of general rules of negotiation and initialise the negotiation process. Once the appropriate negotiation protocol is chosen by the agent, the changeable strategy modules with varying reasoning policies are loaded. The policies contain a set of goals, actions, and rules [Paprzycki et al., 2004].

JADE is a generic agent platform for the implementation of different types of agent applications. Classical JADE agents are behaviour-based, compared to the cognitive types that enable a goal-directed view of agents. For the implementation of cognitive agents,

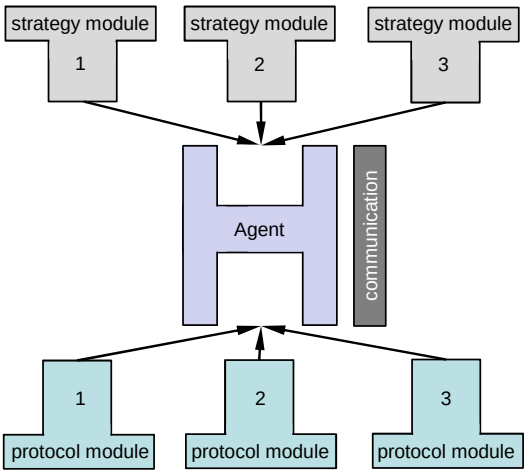


Figure 2.31: Dynamic loading of reasoning models into the generic negotiation agent. Adapted from: [Paprzycki et al., 2004].

literature gives various possibilities, whereof the Belief-Desired-Intention (BDI) paradigm is the most common [Bellifemine et al., 2007]. For the implementation of BDI agents with JADE additional support on the implementation level is required. The Jadex platform is set up as BDI extension for JADE [Braubach et al., 2008, Pokahr et al., 2005], and presented in the following.

Jadex

Jadex platform supports the implementation of cognitive agents in combination with classical software engineering techniques like eXtensible Markup Language (XML) or Java [Braubach et al., 2004], in addition to the compatibility to JADE [Bellifemine et al., 2007]. Cognitive agents are able to handle an explicit representation of their environment, and the reasoning on this representation generates their behaviour. Due to this individual interpretation of messages, strict interaction protocols, as used in classical JADE behaviour-based agents, are dispensable in Jadex [Bellifemine et al., 2007].

Although the BDI paradigm is just one way to model cognitive agents, it is the "most popular" [Bellifemine et al., 2007]. Originally the BDI concept was conceived by Bratman as a "theory of human practical reasoning" [Bratman, 1987]. Since the first investigations in the late 1980s, the strength of this model was the constant use of the folk psychological terms that closely correspond to the way how people communicate about their own behaviour. Related to this initial intention of [Bratman, 1987], the highly influential work of [Rao and Georgeff, 1991] established from then on the consistent use of beliefs, desires,

and plans. These elements are the main components of Jadex as presented in figure 2.32.

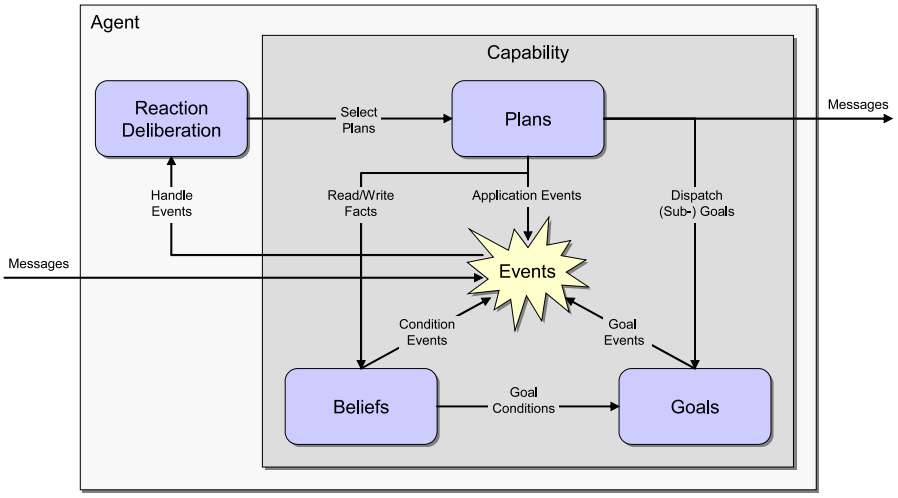


Figure 2.32: Overview of Jadex components. Source: [Pokahr et al., 2005].

As beliefs, goals and plans are central in Jadex, they are further defined in the following:

- Belief: A belief is an access point for data contained in the agent as any kind of Java object. Beliefs are stored in the form of expressions that are dynamically evaluated by agents during runtime.
- Goal: The "concrete momentary desires" [Pokahr et al., 2005] of an agent are its goals that are not necessarily consistent to each other. Due to the set of goals, the agent is directed to (or refrained from) a specific action.
- Plan: A plan constitutes the behavioural element of a Jadex agent and defines the concrete action it carries out. The head of a plan is composed of a condition, and the body of a plan is the actions to take, in order to achieve a goal. [Pokahr et al., 2005].

A behaviour-based agent reacts solely to incoming messages, whereas the Jadex agent handles incoming messages, internal events, and goals by selecting and executing plans (cf. figure 2.32). Compared to the classical approach, Jadex offers a much higher degree of extensibility and flexibility to the resulting agent application. BDI actions can be easily added to the system if desired, and the agent deliberates continuously about its current goals.

With a focus on JADE [Bellifemine et al., 2007] and Jadex [Pokahr et al., 2005], two widely used techniques for the implementation of classical and cognitive agent architectures

have been presented. Due to its exploitation of object-oriented software engineering, a specific application of AOSE methodologies is not required during implementation. Even if the resulting agent concept is different for both platforms, they both rely on a standardised agent communication and the according FIPA standards, as introduced in the next section.

2.3.2 FIPA Standards and Interaction Protocols

The Foundation for Intelligent Physical Agents (FIPA) provides a collection of standards for the interaction of heterogeneous agents and the according agent services. FIPA was accepted in 2005 by the IEEE (Institute of Electrical and Electronics Engineers) as its eleventh standards committee [FIPA, 2011].

The FIPA standards contain twenty-five specifications in total split into five categories: Agent communication, agent transport, agent management, abstract architecture, and applications. As the agent communication is the "core part of FIPA standards" [FIPA, 2011], the focus in this section is set on this category with specifications about

- interaction protocols,
- ACL,
- speech act theory-based communicative acts [Singh, 1991], and
- content language representations [FIPA, 2011].

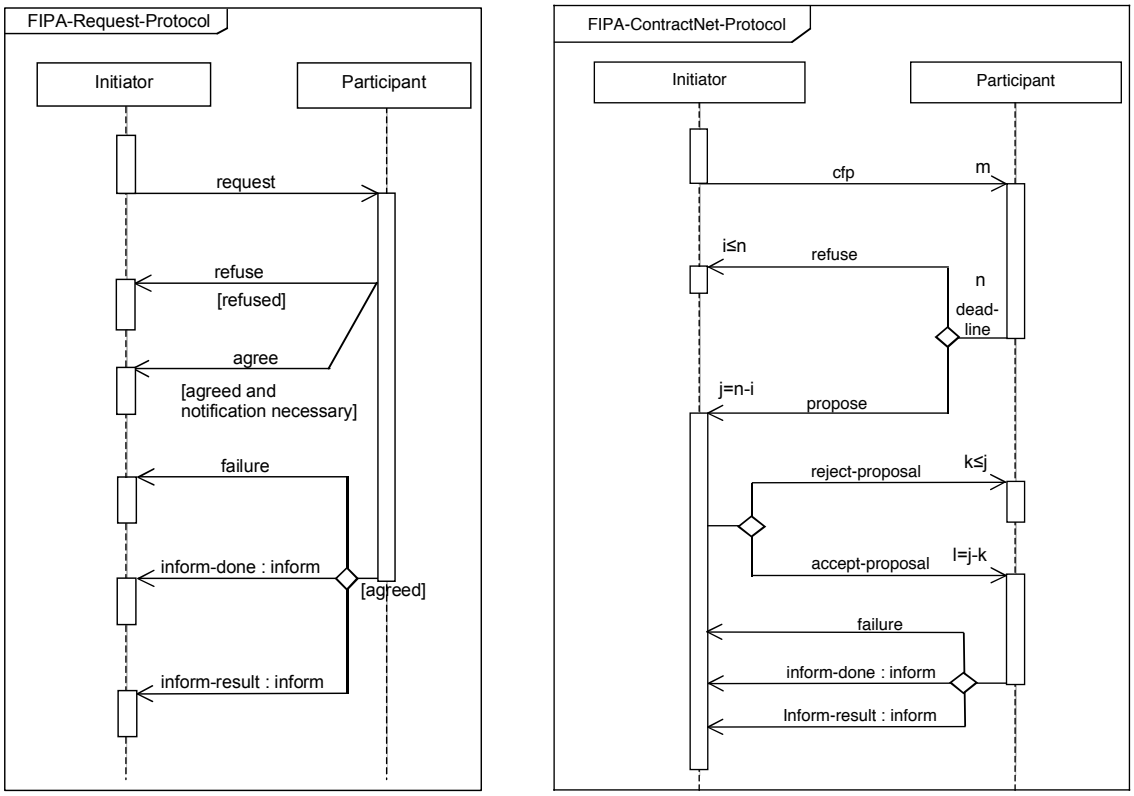
Focusing on the interaction protocols, two of nine available interaction protocols are relevant for this work. The FIPA Request Interaction Protocol and the FIPA Contract Net Protocol (CNP).

The sequence diagram of the FIPA Request Interaction Protocol is illustrated in figure 2.33(a). In this protocol, the Initiator is capable of requesting another agent, the Participant, to perform a desired action. The Participant handles the *Request*-message and decides whether to accept or refuse the request [Bellifemine et al., 2007]. This highly universal interaction protocol can be applied whenever an agent desires another agent to perform a predefined action.

The FIPA CNP is selected as one of the most complex FIPA protocols but with a large number of applications in literature. The investigation of the CNP can be traced back to the 1980s and was initially presented in [Smith, 1980]. Thenceforward the CNP has been widely used in distributed intelligence. In manufacturing, the CNP has first been used for

task distribution among hierarchical organised manufacturing entities in the Yet Another Manufacturing System (YAMS) [Kanchanasevee et al., 1997]. The use of the CNP was continued in the holonic paradigm that exploits the CNP due to its generality [Fletcher et al., 2001], as presented in section 2.2.1.

The Call For Proposal (cfp)-messages, sent from the Initiator to all Participants, are the beginning of the CNP. Thereafter, the Initiator evaluates all received *Propose*- and *Refuse*- messages and in turn sends an *Accept*-proposal to the chosen Participant. The CNP, as given with its protocol flow in figure 2.33(b), is used for generic agent negotiation as it provides the possibility to analyse different proposals of a request with the selection of the most attractive offer.



(a) FIPA Request Interaction Protocol.

(b) FIPA Contract Net Protocol.

Figure 2.33: Sequence diagrams of the FIPA Request Interaction and FIPA Contract Net Protocols. Source: [FIPA, 2005].

Since its first publication, a set of CNP variations were investigated in literature. Besides the Iterated CNP [FIPA, 2005] that allows multi-round iterative bidding, the Concurrent CNP has been investigated for the multi-agent resource allocation (MARA) [Chevaleyre et al., 2006]. The use of the traditional CNP leads to unsatisfactory results

if many Initiators negotiate simultaneously with many Participants. The Participants are required to answer a single bid at a time, consequently they miss some requests. The advantage of the Concurrent CNP is that many negotiations can be conducted simultaneously and, by delaying the final acceptance, better deals can be negotiated.

With the FIPA Request Interaction Protocol and the FIPA CNP, direct request executions and general agent negotiations can be carried out by a MAS and provide a basis for effective agent interaction in MobComm.

The use of AOSE methodologies, their delimitation of object-oriented software engineering, and the provided agent platforms are summarised in the following.

2.3.3 Discussion

In [Wooldridge et al., 2000] it is argued that object-oriented software techniques cannot be applied to effective agent use. The analysis of agent platforms and system implementations in the manufacturing area, presented in section 2.2, leads to the established JADE platform and its extension Jadex. The Java-based platforms provide a FIPA compatible engineering methodology using traditional object-oriented methodologies without the need of AOSE methodologies. The desired implementation of MobComm can thus be conducted with state of the art methodologies of object-oriented software engineering.

2.4 Conclusion

The previous sections reviewed the research areas related to the MobComm objectives. Besides general introduction into mobile robotics, manufacturing systems, and AOSE, each section presented a set of approaches relevant for MobComm with the respective level of objective compliance. According to their presentation in the review, all approaches are assigned to a research area and therein they have a specific scope. Table 2.6 summarises the reviewed areas and gives examples along with the main goals of the selected approaches.

The integration of the MobComm approach in table 2.6 emphasises its association with both the research area of mobile robotics and manufacturing systems. In contrast to approaches like RIA [Guedemann et al., 2006] that considers production cells or the 3T architecture that is a pure mobile robot control, MobComm connects both research areas and adapts solutions and mechanisms of them to solve the given research objectives. Due to the placement of MobComm in table 2.6, the set of MobComm objectives has been

Area	Scope	Literature (examples)	
		Approach (examples)	Main Goal
Manufacturing Systems	Manufacturing control systems	ADACOR [Leitao and Restivo, 2008]	Optimisation of production throughput
	Assembly systems	EAS [Barata, 2006]	Flexible adaptation of assembly systems
	Production cells	RIA [Gudemann et al., 2008]	Reconfiguration after hardware failures and changes in task scheduling
Mobile Robotics	Mobile robot as a manufacturing component	MobComm [Angerer et al.,2010]	Reconfiguration after functional process changes
	Robot control architectures	3T [Firby, 1996], Saphira [Konolige and Myers, 1996]	Flexible control of robot hardware in a dynamic environment

Table 2.6: Scope of related work in manufacturing systems and mobile robotics.

reviewed in the previous sections and summarised in table 2.7.

The implementation of self-organisation (Objective 1) was debated in section 2.2.2 by the presentation of self-organising manufacturing systems. As the self-organising aspect is highly developed in manufacturing systems, EAS [Barata et al., 2006] and RIA [Guedemann et al., 2006] give solutions that can be adapted to the requirements of MobComm. Compared to the implementation of self-organisation, that can be adapted from manufacturing systems to MobComm, the maintenance of productivity during reconfiguration (Objective 1) provides only solution ideas in the holonic paradigm. Even if holonic architectures like ADACOR [Leitão and Restivo, 2008] or PROSA [Van Brussel et al., 1998] provide a basis by the hybrid control architecture, reconfiguration and manufacturing executions cannot be parallelised. Neither is the Meta-Level Components approach [Edwards et al., 2009], taken from the mobile robotics area, able to give a solution with its proposed components for implementation and system monitoring. As the suggested architectural segmentations only serve as a basis for the maintenance of productivity in MobComm, this aspect of Objective 1 contributes to given literature.

The second objective, however, is focused on the dependable integration of new skills into the system. The Organic Computing domain provides solutions by the implementation of an observer/controller architecture in RIA [Guedemann et al., 2006]. In addition to the dependable restoration of invariants, the use of state-chart synchronisation enables high dependability after system reconfiguration in the P'n'P [Naumann et al., 2007] approach. These concepts of Organic Computing and knowledge engineering domain are adaptable

	Objective 1		Obj. 2	Objective 3	
	Self-organised reconfiguration	Maintenance of productivity	Dependability	Reconfiguration after functional changes	Hardware abstraction
3T [Firby, 1996]	No	n/a	n/a	n/a	No
Saphira [Konolige and Myers, 1996]	No	n/a	n/a	n/a	No
MAS4MAR [HaiHua and MiaoLiang, 2007]	Yes	n/a	n/a	n/a	n/a
Meta-Level Components [Edwards et al., 2009]	n/a	part.	n/a	No	n/a
Reconfigurable 3T [Sykes et al., 2008]	n/a	n/a	n/a	No	n/a
SHAGE [Kim et al., 2006]	Yes	part.	n/a	No	n/a
Temporal Logic Motion Planning [Fainekos et al., 2009]	n/a	n/a	Yes	n/a	No
CoBASA [Barata and Camarinha-Matos, 2003]	Yes	part.	n/a	n/a	Yes
Agent-based Commissioning [Staab et al., 2004]	No	n/a	n/a	No	Yes
PABADIS [Feng et al., 2007]	Yes	n/a	n/a	n/a	Yes
PROSA [Van Brussel et al., 1998]	Yes	part.	n/a	No	Yes
ADACOR [Leitao and Restivo, 2008]	Yes	part.	n/a	No	Yes
EAS [Barata et al., 2006]	Yes	part.	No	n/a	Yes
RIA [Guedemann et al., 2008]	Yes	No	Yes	No	Yes
SIARAS [Bengel, 2007]	No	No	Yes	part.	Yes
Plug and Produce [Naumann et al., 2007]	No	No	Yes	n/a	Yes
Ontology-based Reconfiguration Agent [Alsafi and Vyatkin, 2010]	Yes	n/a	part.	No	Yes

Table 2.7: Summary of influential literature and its impact on MobComm.

to MobComm needs. A formal approach to implement dependability in a computation system is provided by the application of Linear Temporal Logic in [Fainekos et al., 2009].

Besides further investigating the maintenance of productivity during reconfiguration, the main contribution of MobComm is its reconfigurability after functional process changes. The required reconfigurability as introduced in the motivation section 1.1 cannot be achieved by any of the presented approaches or architectures. Only the knowledge-intensive SIARAS [Bengel, 2007] approach can be used as a starting point for functional reconfigurability but lacks in the implementation of self-organisation and the maintenance of productivity. Reconfigurability in manufacturing research is mostly focused on hardware failures, task scheduling (e.g. RIA [Guedemann et al., 2006]), or production through-

put changes (e.g. ADACOR [Leitão and Restivo, 2008]). Even if these approaches like ADACOR [Leitão and Restivo, 2008] or RIA [Guedemann et al., 2006] implement a progressive reaction to system-internal events like hardware failures, they do not provide the framework to adapt a mobile robot to functional process changes without programming effort. The methodologies of hybrid system theory include a domain-independent mechanism for robot behaviour verification but lack in hardware abstraction and thus in an easy change of robot components.

The type of reconfigurability proposed in this thesis can be applied to standalone manufacturing components or integrated into manufacturing control systems with established reconfigurability after hardware failures or production load changes. The integration into existing manufacturing approaches has been elaborated with ADACOR [Leitão and Restivo, 2008] in figure 2.19 on page 43 and RIA [Guedemann et al., 2006] in figure 2.25 on page 54. The novel reconfigurability after functional process changes at component level contributes to an enhanced flexibility of the used manufacturing control system.

The second part of Objective 3 (cf. section 1.3.2 on page 9) contains hardware abstraction. This characteristic in not regarded or not provided in mobile robotics approaches but state of the art in manufacturing systems research as emphasised by the large set of complying approaches in table 2.7. The idea of an encapsulated resource agent layer as given in many approaches such as CoBASA [Barata and Camarinha-Matos, 2003] or PABADIS [Klostermeyer and Klemm, 2003] is widely used in literature and transferable to MobComm to fulfil hardware abstraction in this work.

Section 2.3 reviewed the AOSE techniques for system implementation. As Java-based agent platforms like JADE [Bellifemine et al., 2007] or Jadex [Braubach et al., 2004] and FIPA Agent Communication standards are provided, state of the art techniques, based on object-oriented system engineering, allow the MobComm implementation without refraining the merits of agent technology.

After reasoning on the contribution of this work, a set of research tasks is deduced for further investigation of the reconfiguration mechanisms and its evaluation in the following chapters. The key contributions of MobComm, as discussed above, are mapped into a set of research tasks, summarised in table 2.8.

These research tasks reflect all research objectives (Task 1 - Task 8) and require a self-organised, dependable, and hardware-abstract functional reconfiguration mechanism that is maintaining productivity. The compliance of these tasks that focus on the reconfi-

		Task description	Task number	Chapters in thesis
Research tasks	Objective 1	Provide a reconfiguration mechanism that realises self-organisation.	Task 1	Reconfiguration mechanism and Validity Check (chapters 4 and 5)
		Provide a reconfiguration mechanism that does not affect the level of productivity during reconfiguration.	Task 2	
	Objective 2	Provide mechanisms that ensure dependability in the use of new functionalities.	Task 3	
	Objective 3	Provide a reconfiguration mechanism that allows hardware abstraction.	Task 4	
		Provide a reconfiguration mechanism that is robot configuration independent.	Task 5	
		Provide a reconfiguration mechanism that is aware of the limitations of its reconfiguration capabilities.	Task 6	
		Provide a reconfiguration mechanism that is open for a broad range of functional process changes.	Task 7	
		Provide a satisfactory fast adaptability to new processes.	Task 8	
Supportive tasks	Objective 1	Provide a system architecture that supports self-organisation.	Task 9	System architecture (chapter 3)
		Provide a system architecture designed for the maintenance of productivity.	Task 10	
	Objective 2	Provide a system architecture for the dependable integration of new skills.	Task 11	
	Objective 3	Provide a system architecture that allows functional reconfigurability.	Task 12	
		Provide a system architecture that allows hardware abstraction.	Task 13	
	Total system	Use of software engineering methodologies that allow the compliance of Objectives 1-3.	Task 14	Implementation (chapter 6)
		Use of software frameworks and tools that produce measurable and comparable outcomes.	Task 15	

Table 2.8: List of research and supportive tasks sorted by the related research objectives.

guration mechanism is the main goal of the present thesis. System architecture is regarded as a supportive set of tasks (Task 9 - Task 13) as it facilitates the realisation of the reconfiguration mechanism. The supportive tasks 14 and 15 emphasise the relevance of system implementation and evaluation in this thesis. The tasks numbered in table 2.8 are used as a design and implementation guideline in chapter 3 to 5, especially the evaluation chapter of this work is based on the level of task compliance provided by the MobComm implementation.

After the review of related research areas, the analysis of relevant approaches, and the presentation of a list of research tasks, the following chapter focuses on the investigation of the MobComm architecture.

Chapter 3

Design of MobComm Architecture

The proposed skill-based reconfiguration, as introduced in section 1.5, is presented in chapter 3 to chapter 5.

In the following, however, the MobComm architecture is explained with reference to the supportive tasks given in table 2.8 on page 74 (Tasks 9-13). Their compliance is the guideline for the presented architecture that aims to ensure the desired characteristics. Only an architecture design that permits self-organisation (Task 9) and maintenance of productivity (Task 10) while allowing functional reconfigurability (Task 12) and hardware-abstraction (Task 13) is able to support the investigated reconfiguration mechanism efficiently.

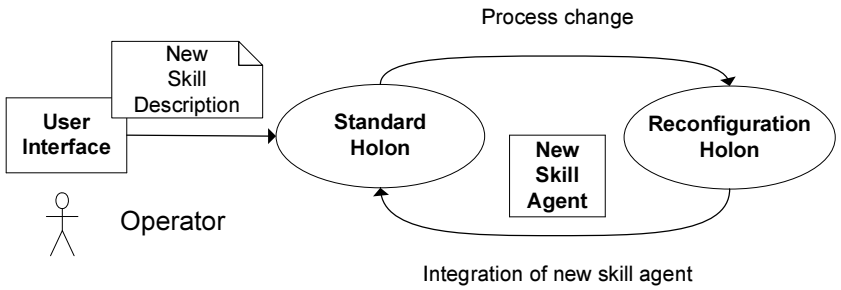


Figure 3.1: Overview of MobComm architecture.

Figure 3.1 introduces the architecture components used. The total architecture is divided into holons - a Standard Holon (SH) and a set of Reconfiguration Holons (RH). Within the scope of this thesis the creation of a single Reconfiguration Holon is focused. The Standard Holon is responsible for the execution of routine tasks and provides the

interfaces to the robot hardware and the application environment. If the functional process requirements have been changed by a human operator or a superior manufacturing system, an instance of a Reconfiguration Holon is initiated to perform solely reconfiguration tasks. The presented architecture allows to encapsulate reconfiguration activities and results in the provision of a new robot functionality in the form of a new Composite Skill Agent.

Even if the term Skill is discussed and defined in section 3.3, a distinction between Task, Skill and behaviour is introduced at this early stage. A Task describes a sequence of Skills that differ either in their service names or in their parameters like component names in case the service names are identical. In contrast to that, a Skill always includes a unique robot functionality that exceeds a simple parametrisation of already existing Skills. A resulting system behaviour, however, is the effect of the executed Skills in real-world combined with the low level control mechanisms of the mobile robot.

In this chapter, section 3.1 introduces the holonic structure, followed by the description of the layered communication hierarchy in section 3.2, and the skill-based design in section 3.3. Before the conclusion is drawn, section 3.4 presents the provided MobComm interfaces.

3.1 Holonic design

The architecture presented is influenced by the holonic principle, as discussed in the literature review in section 2.2.1. This principle has also been established in approaches such as PROSA [Van Brussel et al., 1998], ADACOR [Leitão and Restivo, 2008], or the Evolvable Assembly Systems [Barata et al., 2006].

Due to the advantages of a holonic design, MobComm architecture follows the paradigm that "every item is a whole as well as a part of a bigger whole" [Leitão and Restivo, 2008]. This statement includes that the holon is an autonomous entity itself but can further be part of a control hierarchy without loosing its autonomy.

As presented in figure 3.2, MobComm is the total and the Standard Holon is its part but at the same time an enclosed entity by itself. The Standard Holon constitutes an entire multi-agent system that includes the currently running configuration of the robot such as a commissioning process (cf. figure 1.3 on page 4). The Standard Holon in turn is split into layers that contain agents as overviewed in section 3.2 in figure 3.4.

The second part of MobComm is the set of Reconfiguration Holons. A Reconfiguration

Holon consists solely of agents and thus contains no further internal recursive structure. The interface between Reconfiguration and Standard Holon is characterised by initiation and finalisation of the reconfiguration mechanism as detailed in chapter 4. Additionally the Validity Check that is introduced in chapter 5 utilises the interface between Reconfiguration and Standard Holon.

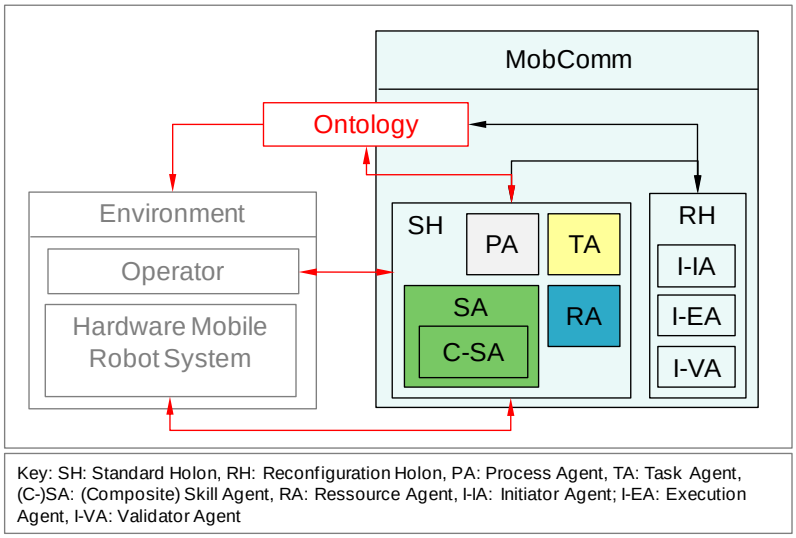


Figure 3.2: Overview of the holonic design in MobComm architecture.

The Reconfiguration Holon is only an optional part of the system and only activated in case a reconfiguration is required after functional process changes. In contrast, the Standard Holon with its standard process execution is the core part of the architecture and contains different types of permanent agents. These agents can be atomic parts or recursive structures as implemented in the Composite Skill Agents (C-SA). The composite agents that are still displaying a recursive agent structure, always resulting from a reconfiguration process. From outside the agent, no distinction between atomic and composite agents can be detected as only dynamic services are available. Even if MobComm does not show a continuous holonic design, its architecture displays the holonic principle most notably in Standard and Reconfiguration Holon besides the Composite Skill Agents.

The interface of MobComm to its environment is presented in figure 3.2. The mapping of system vocabularies by an ontology, as seen in section 2.2.3, and the interface to the environment are managed by Standard Holon. As shown in figure 3.2, a Reconfiguration Holon and the application environment are only connected by the system ontology as introduced in section 3.4.

The total encapsulation of reconfiguration activities with a defined interface between the interacting holons is a core goal of this thesis as introduced in chapter 1. Compared to approaches like ADACOR [Leitão and Restivo, 2008] or RIA [Guedemann et al., 2008] that execute their reconfigurations without a parallel standard process, the goal of MobComm and the named approaches differs in the reconfiguration events they can handle. ADACOR [Leitão and Restivo, 2008] or RIA [Guedemann et al., 2008] deal with disturbances and hardware breakdowns as reconfiguration impulses. The process changes that cause functional reconfigurability in MobComm do not affect the actual running process and thus allow the maintenance of productivity by a complete encapsulation of reconfiguration activities.

By focusing on the internal composition of the holons instead of an overview of the architecture, a set of different compositions are provided as shown in figure 3.3.

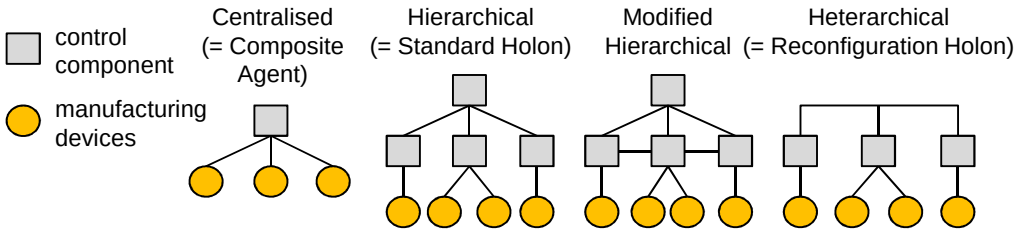


Figure 3.3: Overview of agent structures with the association of MobComm holons. Adapted from: [Dilts et al., 1991].

Standard Holon includes a hierarchy for the required robustness and dependability of standard process execution, on the other hand a heterarchy is implemented in Reconfiguration Holon that gives flexibility for a self-organised reconfiguration mechanism. A centralised communication hierarchy is used in the Composite Skill Agents that are created during the reconfiguration process. Figure 3.3 overviews a set of possible agent structures based on [Dilts et al., 1991] with the association to the introduced MobComm components.

The internal structure of the Reconfiguration Holon and Composite Skill Agents is detailed in chapter 4 due to its relation to reconfiguration activities. The communication hierarchy of Standard Holon provides the basis for these reconfiguration activities and is described in the following section.

3.2 Standard Interaction Hierarchy

This section presents the interaction hierarchy of Standard Holon with the exclusion of reconfiguration activities as they are examined in chapter 4. Besides the presentation of the applied agent layers and the contained agent types, interaction mechanisms of Standard Holon are presented in the following section.

The layers specific for MobComm architecture are presented in figure 3.4 along with robot control and hardware layers. Control and hardware layers, however, must be provided by the robot manufacturer to the car manufacturer and they are assumed as given in this thesis. The provided Application Programming Interface (API) of the used robot control is mapped in the system's Resource Layer as described in section 7.3.

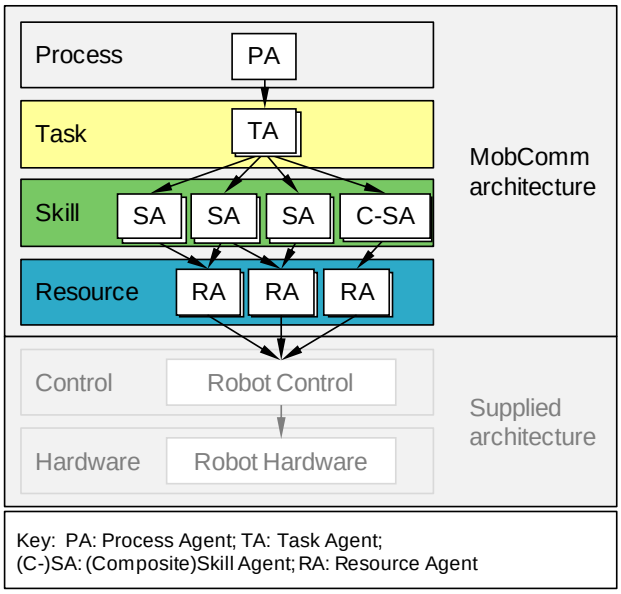


Figure 3.4: Overview of agent, control, and hardware layers in MobComm architecture.

Process and Task Layers are responsible for the decomposition of manufacturing processes whereas Skill and Resource Layers constitute the provision of the robot configuration to the decomposed manufacturing process. The basic concept of skills is taken from 3T architecture [Gat, 1992] as discussed in section 2.1.1 on page 17 where the set of enabled skills creates the resulting system configuration. All skills are independent of each other and describe the robot-specific interface to the world [Gat, 1992]. This assumption is set in this thesis (in addition to those assumptions set in section 1.3.3) as the encapsulated reconfiguration mechanism must be able to regard the set of already existing skills as independent of each other. This independence is the prerequisite for the encapsulated

execution of the distributed skill composition as described in section 4.2.

The Resource Layer represents the connection of the mobile robot to the real-world environment corresponding to the robot hardware as proposed in section 2.4 as a solution for the desired hardware abstraction in MobComm (Task 12). A Resource Layer is state of the art in agent-based manufacturing approaches such as PABADIS [Feng et al., 2007] or PROSA [Van Brussel et al., 1998].

Following the hierarchy of agent layers presented in figure 3.4, a corresponding type of agent is introduced for each layer. Instances of Process Agent (PA), Task Agent (TA), Skill Agent (SA), and Resource Agent (RA) form the corresponding MobComm layers. An example for the decomposition of a manufacturing process in MobComm Layers is given in the motivation section 1.1 and more detailed in section 3.3.2.

The Skill Layer, as the architecture's core level, is divided into Atomic Skill Agents, mapping the basis functionalities of the used mobile robot system, and Composite Skill Agents (C-SA) that emerge after a MobComm reconfiguration. Compared to the atomic agents, composite ones follow the holonic principle as evaluated in the last section. Both agent concepts provide an identical activation and termination behaviour. Figure 3.5 summarises the functionalities of MobComm layers and their agents. In addition to that, figure 3.5 specifies the interaction mechanisms between the different agent layers.

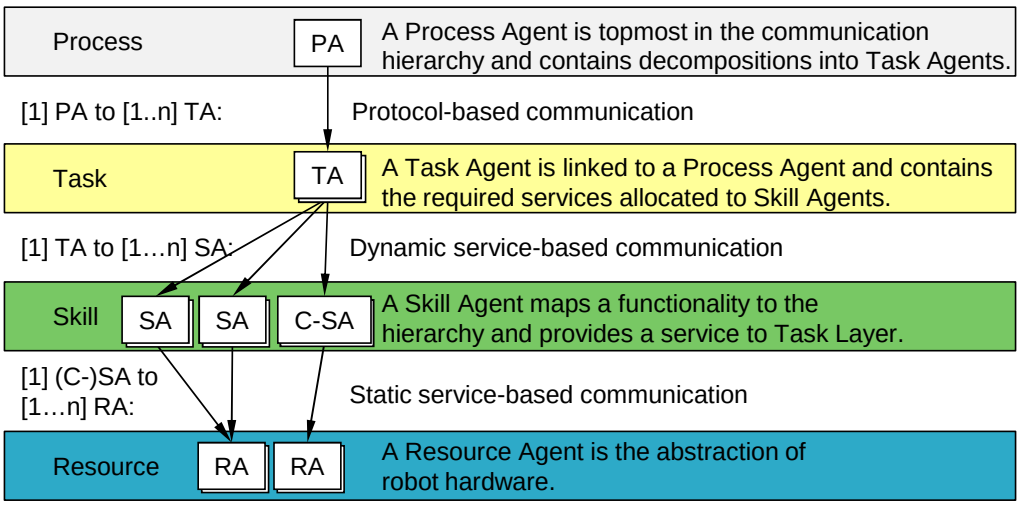


Figure 3.5: Agent types and interaction in Standard Holon.

A single Process Agent communicates in a protocol-based manner with a set of Task Agents with the result of a planned commissioning process. This mechanism utilises the FIPA Request protocol [FIPA, 2001] and works monodirectionally from the Process

Layer to the set of Task Agents. The interaction activities between Task and Skill Layer however are service-based. A single Task Agent requires services from Skill Layer without a reference to a specific Skill Agent. This interaction is dynamic as the set of possible services is not limited and can be extended during runtime. An example service allocation by a Task Agent is illustrated in the extracted communication protocol in figure 3.6.

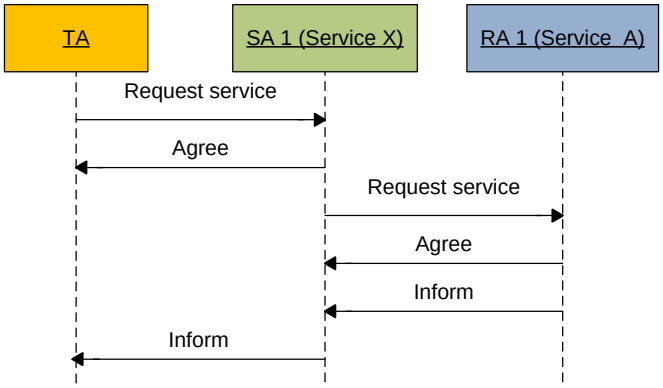


Figure 3.6: Example interaction between Task, Skill, and Resource Layer.

Even if the interaction between Skill and Resource Layer is service-based as well, it is static compared to the service allocation of Task Layer. It is only possible to dynamically allocate but not to dynamically change services during runtime. Even if a single Skill Agent can allocate more than one Resource Agent, only a single Resource Agent can be allocated by a Skill Agent at a time. The problem of an efficient resource allocation is beyond the scope of this thesis whereas a possible solution is given in the survey of [Chevaleyre et al., 2006] about multi-agent resource allocation (MARA).

Whereas this section specified the general functionalities of Standard Holon-agents, the integration of planning and scheduling into the system is elaborated in the skill-based design of the next section.

3.3 Skill-based Design

Based on the descriptions of agent layers and types in figure 3.5, the integration of planning and scheduling in MobComm architecture is detailed in this section. For the further description of scheduling and planning activities, the terms Task and Skill are defined in the following in a general manner:

Definition 3.1 (General Skill) *A Skill is an elementary sensor-based robot movement, like MoveTo, a system command, like OpenGripper, or a sensor function, like LocateObject [Mosemann and Wahl, 2001].*

Definition 3.2 (General Task) *A Task is an activity whose execution may require obtaining several services from an environment as well as accessing several materials [Sousa et al., 2006].*

The MobComm architecture requires the commissioning process to be planned and scheduled for the execution of a decomposed process on the robot hardware dynamically. For a clear understanding throughout this work both terms are defined subsequently:

Definition 3.3 (Planning) *Planning selects and sequences activities such that they achieve one or more goals and satisfy a set of domain constraints [Fox, 1994].*

Definition 3.4 (Scheduling) *Scheduling selects among alternative plans, and assigns resources and times for each activity so that the assignments obey the temporal restrictions of activities and the capacity limitations of a set of shared resources [Fox, 1994].*

Planning and scheduling are essential for the accomplishment of complex activities with mobile robots. Planning decides which activities to perform while scheduling provides information about the temporal order and the required resource allocation in a process [Cass et al., 2001]. In order to be able to reconfigure given robot functionalities for manufacturing processes in MobComm, planning and scheduling have to be applied consistently. As stated in the research assumptions in section 1.3.3 on page 10, the definition of a specific planning algorithm is not in the scope of MobComm and assumed to be provided. Scheduling, however, with its temporal and conditional aspect is regarded as inseparable from the requirement of functional configuration. According to that, two possibilities to distribute scheduling in Skill and Task Layer are discussed in the following section.

3.3.1 Scheduling Distribution

The definition of scheduling in the last section demands the integration of temporal and conditional aspects that can either be integrated in Task or Skill Layer as well as distributed among both. The skill-based and task-based approaches are evaluated subsequently towards their compliance of Research Tasks listed in table 2.8 on page 74.

The task-based approach as given in figure 3.7 is the first possibility to implement scheduling.

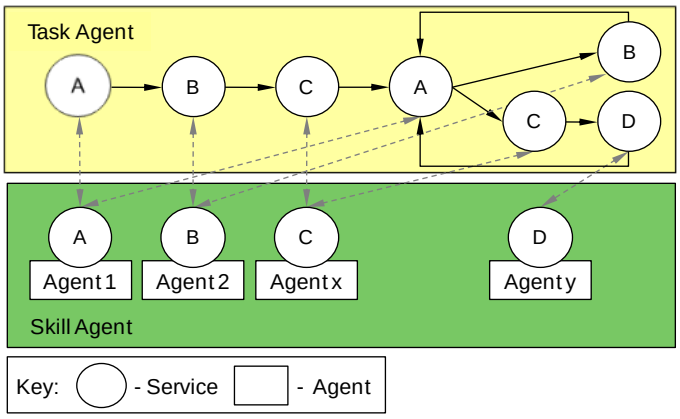


Figure 3.7: Overview of example scheduling in the task-based approach with scheduling knowledge centralised in Task Layer.

There, the Task Layer includes the complete scheduling information about conditional and temporal allocations of skills. Temporal and conditional scheduling is centralised in Task Layer whereas the knowledge of task structure is additionally provided by the operator. Typically, this task structure is a linear or conditioned sequence of system skills that maps the requirements of the manufacturing process. Consequently, the required self-organisation (Research Task 1) must be integrated in Task Layer and would allow a self-organised assignment of skills.

In contrast to the task-based approach, the skill-based design, as provided in figure 3.8, distributes scheduling activities between Skill and Task Layers. Whereas the management of the temporal aspect remains in Task Layer, the conditional aspect is integrated in Skill Layer. This implicates that temporal scheduling knowledge is provided by the operator and the conditional complexity is hidden in the dynamic provision of services by skill layer. The individual robot functionalities can be inserted once by the user and reused without further effort.

The integration of self-organisation in Skill Layer that manages reconfiguration of robot functionalities decentralised and without outside control (cf. definition self-organisation 2.12), allows an encapsulated handling of functional process changes as desired in Research Task 7.

The proposed MobComm architecture has to comply with the defined tasks whereas for the scheduling integration in MobComm architecture only a set of Supportive Tasks is identified as relevant:

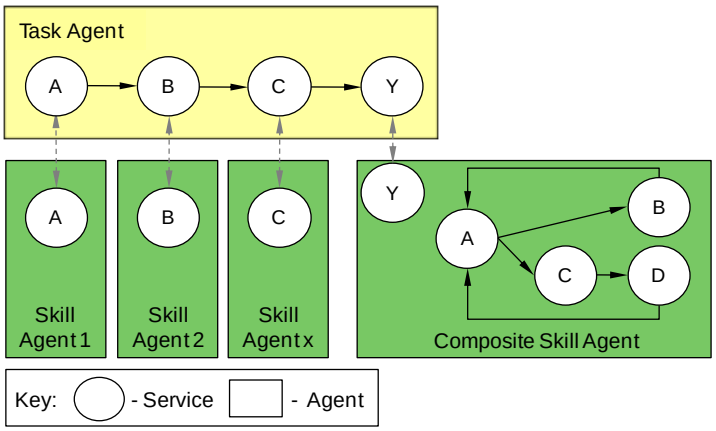


Figure 3.8: Overview of example scheduling in the skill-based approach with scheduling knowledge distributed among Task and Skill Layer.

- Support of self-organisation (Task 9) and
- integration of functional reconfigurability (Task 12).

For the evaluation of both approaches, table 3.1 gives an overview of the single task compliances. Both approaches comply with the required support of self-organisation (Task 9) even though in different layers. As described above, the skill-based concept demands its implementation in Skill Layer whereas a self-organised Task Layer is provided in the task-centric approach.

The effort to integrate a classical planning mechanism to schedule the corresponding Tasks as proposed in [Nau et al., 2004] is required in both approaches. Due to the industrial requirement of a robust standard process execution as introduced in section 1.1, no learning mechanisms are integrated in Task or Skill Layer of MobComm.

The most relevant difference between the proposed designs is the integration of functional reconfigurability (Task 12). As discussed in the literature review in section 2.4, functional reconfigurability is one of the main contributions of this work. Only the skill-based design allows new robot functionalities to be integrated self-organised and encapsulated in the already running system.

By analysing the results of table 3.1, the possibility to integrate functional reconfigurability is the basis to decide upon the application of a skill-based MobComm architecture as given in the example scheduling in figure 3.8.

The assumption that the task sequence has to be linear in MobComm is set to clearly differentiate between the two presented approaches in this thesis. For a future extension of

	Task-based approach	Skill-based approach
Self-organisation (Task 9)	Yes	Yes
Functional reconfigurability (Task 12)	No	Yes
Task planning required	Yes	Yes
Learning mechanism included	No	No

Table 3.1: Comparison of skill-based and task-based approaches.

MobComm a conditioned Task Layer is desired to additionally benefit from higher planning capabilities in Task Layer without loosing the advantage of functional reconfigurability in Skill Layer.

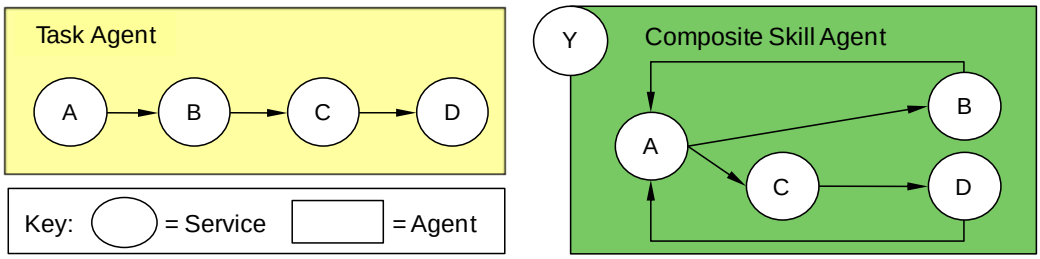
Based on the general definitions 3.1 and 3.2, MobComm Skills and Tasks map the integration of the skill-based design in the following definitions:

Definition 3.5 (MobComm Task) *A MobComm Task is a temporal specification considering the allocation of services. Task Agents contain only temporal scheduling information and the according services of required Skill Agents.*

Definition 3.6 (MobComm Skill) *A MobComm Skill is a unique capability related to the functionality of a mobile robot system. A set of Atomic Skill Agents provides the basic functionalities of the robot while Composite Skills contain conditional scheduling activities. The interface to a MobComm Task is the dynamic provision of specific services that are registered at a central management system.*

Based on the given definitions, Task and Skill Agents, as pictured in figure 3.9, differ in structure, agent activation, and repetition behaviour. As summarised in table 3.2, a Task is composed of a linear sequence, where a Skill contains complex structures and uses a simple service-based interface to Task Layer. In contrast to a MobComm Skill that can be repeated by condition-based events, a Task is only executed after an afresh activation by Process Layer. The use of the Composite Skill Agents in future MobComm reconfigurations is directed to section 4.4.

Independent of the applied scheduling concept, a parallel execution of agents in Skill or Task Layer is only applicable with restrictions. Due to the set requirement that all skills have to be independent of each other (cf. page 79) and the given disconnection of robot control and MobComm architecture, Skills can only be parallelised if the robot control provides the according functionality in its API. The mitigation of this MobComm



(a) Example MobComm Task structure.

(b) Example MobComm Skill structure.

Figure 3.9: Example structures of MobComm Skills and Tasks.

limitation is directed to future work. The MobComm Tasks, however, can be executed in parallel as long as only one task is hardware-related.

	MobComm Skill Agent	MobComm Task Agent
Structure	Complex internal structure, external view as a single service	Linear sequence of services
Activation	By a Task Agent	By a Process Agent
Repetitions	Condition based repetitions (hardware input, environment)	Only after afresh activation
Parallel execution	Applicable if provided by the robot control	Applicable if only one is hardware related

Table 3.2: Description of MobComm Tasks and Skills.

An example of the integration of scheduling in an automotive commissioning process with MobComm is presented in the subsequent section followed by the MobComm architecture interfaces in section 3.4.

3.3.2 MobComm Planning and Scheduling

To emphasise the integration of planning and the division of scheduling in MobComm architecture, the overview of an example commissioning process is detailed in figure 3.10 and explained in the following.

A commissioning process is initialised by an operator that is a human or a superior manufacturing system. An example integration of MobComm in state-of-the-art manufacturing systems like ADACOR [Leitão and Restivo, 2008] (cf. figure 2.19, page 43) or RIA [Guedemann et al., 2006] (cf. figure 2.25, page 54) is presented in the literature review. The named approaches act as operators in the presented example.

Process changes are integrated by the operator in MobComm whereas only process

changes executed by a mobile robot are handled in this thesis. Activities that are executed statically or manually are not relevant and given as blue-dashed boxed in figure 3.10. In this example, three processes are planned by the operator: The semi-automatic handling of component Y (Process 3), a fully automated handling (Process 2), and a manual handling (Process 1). The manual handling and the fully automated one, given in grey font, are not related to the mobile robots. Thus, only the semi-automatic handling is relevant as the mobile robot complies with this process in parts.

While focusing on Process 3, three Tasks are implemented by the operator: The manual refill of components (Task 3) and package handling (Task 2) outline the manual part of the semi-automated process. The automated part is the provision of a component in assembly order (Task 1) and is in the further focus in figure 3.10.

Every MobComm Task contains a scheduling information including its temporal constraints and the required sequence of services. Services are allocated by Task Layer from Skills whereas a missing service during the allocation process activates the MobComm reconfiguration mechanism. In the example of figure 3.10, all desired services - Move, Grip, Deposit, and Detect - are provided by Skill Layer as Atomic Skills. Thus, no reconfiguration activities are required for the automated pick-and-place in commissioning.

In this thesis planning, task scheduling, and the definition of services refer to knowledge provided by the operator whereas the Skill provision and resource allocation is generated within the MobComm system. The further development of MobComm architecture towards further self-organised layers is part of future work in chapter 8 and beyond the scope of this work. The flexible provision of services and their execution by the mobile robot system is the main focus of the MobComm reconfiguration mechanism as given in chapter 4 after the presentation of architecture interfaces in the following.

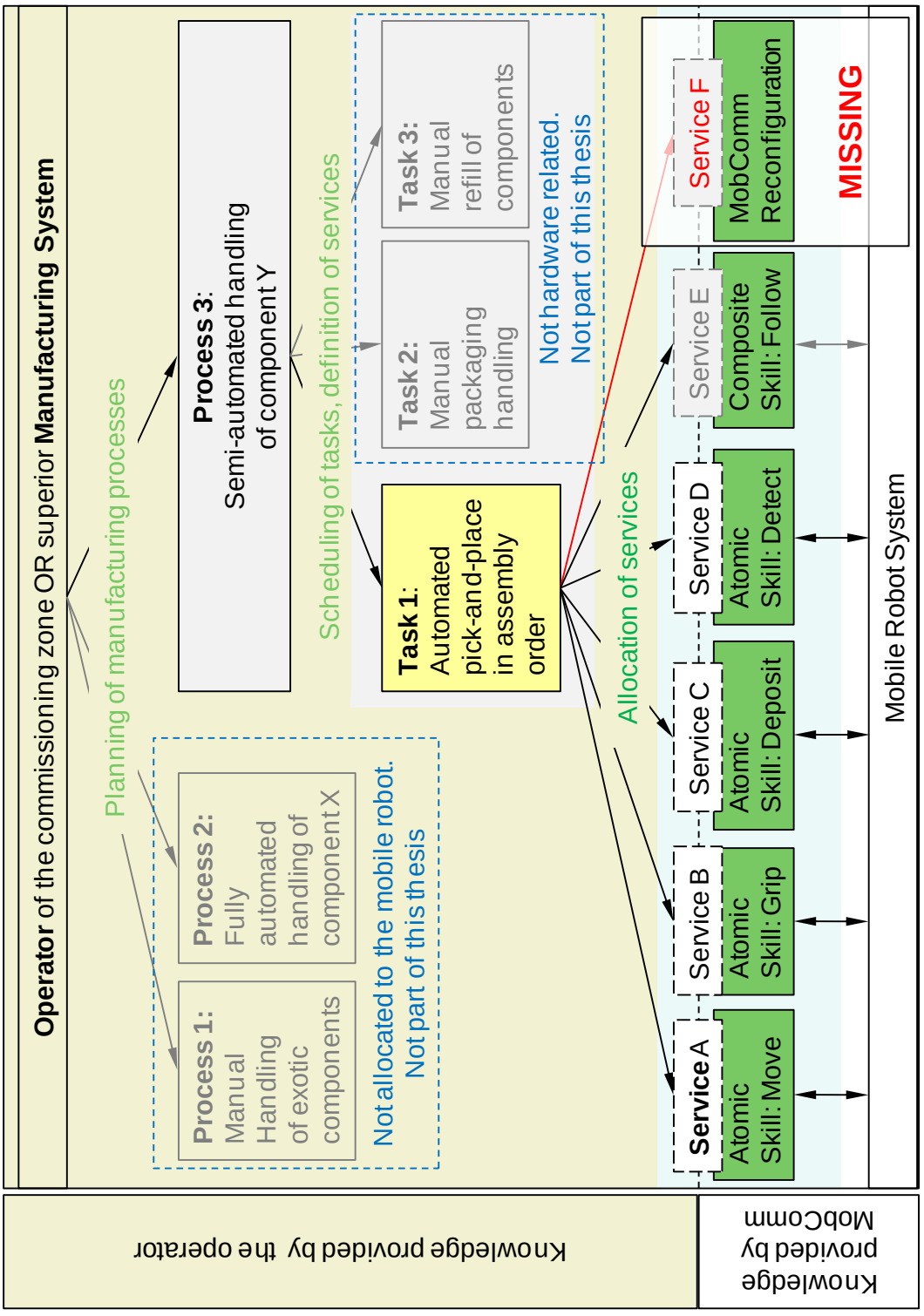


Figure 3.10: Integration of planning and scheduling in automotive commissioning using MobComm.

3.4 Interfaces

While section 3.1 and section 3.2 explained the basic structure and functionalities of MobComm architecture components, this section focuses on the interfaces provided. According to figure 3.2 on page 77, the interfaces to the environment are divided into three parts. An ontology provides the semantic basis for internal or environment-related interactions while further interfaces are given between the Standard Holon and the operator besides the Standard Holon and the robot control. The interaction between Standard and Reconfiguration Holon is given in the next chapter due to its focus on the reconfiguration mechanism.

By following the review of knowledge engineering in manufacturing in section 2.2.3, a semantic representation that is divided into "similar structured partitions" [Schreiber et al., 1999] is important for an adequate knowledge engineering in a system. Literature such as [Frei, 2010, Schreiber et al., 1999, Alsafi and Vyatkin, 2010] proposes the use of an ontology as a flexible and shareable way to integrate semantic information into a manufacturing system (cf. section 2.2.3).

MobComm ontology, as presented in figure 3.11, publishes system vocabularies to the operator and gives the common understanding of them between Standard and Reconfiguration Holon to support reconfiguration activities. The ontology is partitioned into an architecture, an operator, an environment, and an internal part.

Semantics needed for the description of the environment are mapped in the environment section as these are required for the inserted process description. In this thesis, the environment descriptions are modelled as static parts of the ontology whereby its flexibilisation is part of future work in section 8.2. *Application Descriptions* are available to describe the according commissioning applications: The *Location* of the mobile robot, the *Position* of the robot arm in space, and an *EnvObject* to describe an environmental object to be handled, avoided, or detected by the mobile robot, are also available.

To map the hierarchy of Standard Holon in the ontology, every *ResourceDescription* is linked to at least one *SkillDescription*. Whereas a *SkillDescription* is used by at least one *TaskDescription*. The topmost concept in the ontology is a *ProcessPlan*, containing all basic planning and temporal scheduling information. This structure follows the segmentation in the example commissioning process of figure 3.10.

The set of *ApplicationDescriptions*, as described above, is used by *AgentActions* that

represent the effect of the agent executions on the environment. The connection between *ApplicationDescriptions* and *AgentActions* outlines the interface between the environment and the MobComm internal part of the ontology. Skill and Resource Agents have *AgentActions* that require a set of *Preconditions* and *Postcondition* to describe their activation and termination behaviours as detailed in chapter 4. The MobComm internal part further includes the interface to the operator that is implemented by the connection between a *TaskDescription* and a *SkillDescription*.

The core concept of the operator part is the *New Skill Description (NSD)* that constitutes the semantic interface from the operator to the system's ability to reconfigure. In case a process change requires the provision of a new service, the concept *New Skill Description* is used to map the desired functionality into the system.

The third interface of MobComm affects the underlying hardware. As hardware abstraction is Research Task 13, a dynamic interface is provided. Due to the close relation of this work to the setup of the according mobile robot prototype, given in section 7.3, the real-world interface focuses on the interaction with the prototype. Even if the complete hardware abstraction requires enhancements in future work, the design of the MobComm Resource Layer provides basic premises for hardware abstraction. By providing a MobComm Resource API to mobile robot manufacturers, the according Resource Agent can be integrated seamlessly in the system. The communication between Skill and Resource Layer is designed service-based, and thus not affected by changed hardware. Similar to the flexibilisation of Skill Layer, as detailed in section 3.3.1, the introduction of dynamic service allocation in Resource Layer is part of future work. A basis for the resource allocation flexibilisation can be provided through MARA [Chevaleyre et al., 2006].

This section described the interfaces with the environment and the robot hardware, and concludes the presentation of the MobComm architecture.

Holonic structure, agent functionalities, and scheduling design have been focused while they are further discussed towards their compliance of the research tasks in the following conclusion of this chapter.

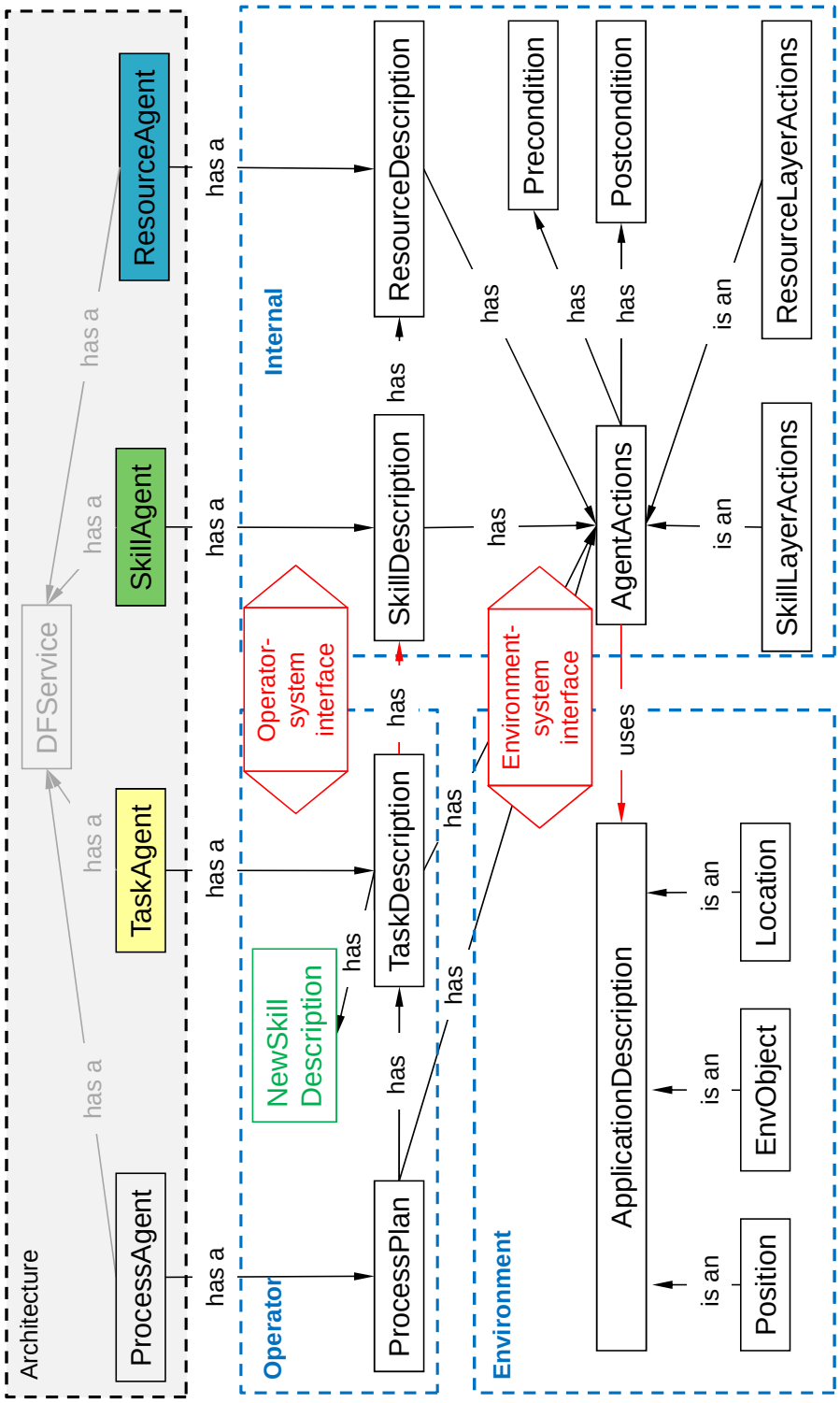


Figure 3.11: Overview of MobComm ontology divided into architecture, operator, environment, and internal parts.

3.5 Conclusion

The presented MobComm architecture is mainly influenced by the holonic manufacturing as a subtype of the agent-based paradigm. The division between reconfiguration and standard process executions, applied in Holonic Manufacturing Systems, is reused in this work. A complete encapsulation of reconfiguration activities is achieved for the MobComm architecture by the application of a Standard and a Reconfiguration Holon. In Standard Holon, Resource, Skill, and Task Layers base on the classical 3T architecture [Gat, 1998] as presented in section 2.1.1. The concentration on Skill Layer within the provided architecture allows the integration of a high level of self-organisation in combination with a dynamic provision of robot functionalities in runtime.

To discuss the task compliances required from MobComm architecture, the set of supportive tasks listed in table 2.8 on page 74 is evaluated in the following with its summary provided in table 3.3.

Supportive Task	Compliance
Provide a system architecture that supports self-organisation (Task 9).	Yes
Provide a system architecture designed for the maintenance of productivity (Task 10).	Yes
Provide a system architecture for the dependable integration of new skills (Task 11).	No
Provide a system architecture that allows functional reconfigurability (Task 12).	Yes
Provide a system architecture that allows hardware abstraction (Task 13).	Yes

Table 3.3: Task compliances of MobComm architecture.

The first task desires the system architecture to support self-organisation (Task 9) and is in compliance with the proposed architecture. As discussed in literature review in section 2.2.2, the use of agent technology enables the provision of self-organisation due to definition 2.9. The holonic design further enables the encapsulation of self-organisation into Reconfiguration Holon. In Standard Holon, however, self-organisation is not desired to maintain productivity at a high level. The heterarchical structure in Reconfiguration Holon prepares the implementation of the self-organised reconfiguration mechanism. As the goal is the application of functional reconfigurability, the proposed architecture provides a self-organising Skill Layer combined with a skill-based scheduling mechanism.

The conflict between the maintenance of productivity (Task 10) and the integration of

self-organisation (Task 9) is solved by complete encapsulation of the reconfiguration mechanism in Reconfiguration Holon from the hierarchical process execution in Standard Holon. Thus, the productivity of the running process can be maintained on a pre-reconfiguration level during its execution. The used service-based interaction between Tasks and Skills supports the level of productivity as the integration of new Skills are realised dynamically without interrupting of the running process.

The dependability of the new Skills, as desired in Task 11, cannot be achieved through the presented architecture. No specific arrangements support dependability during Skill integration. On the contrary, the use of self-organisation in the Reconfiguration Holon with the integration of its results in the running process even decreases the resulting dependability. While the dynamic allocation of services in Standard Holon allows the maintenance of productivity (Task 10), it decreases the level of dependability of Skill integration (Task 11). To react to this lack, a Validity Check is introduced in chapter 5 to meet the requirements of Task 11.

The compliance of functional reconfigurability (Task 12) is achieved by different mechanisms. The provision of atomic robot functionalities in a separate Skill Layer is the basis for this type of reconfigurability. Atomic functionalities are e.g. Move, Detect, Grip, or Manipulate. Enhanced through implementation of conditional scheduling, Skill Layer becomes the central layer of the architecture, and lays thus basis for a self-organised provision of new robot functionalities in the reconfiguration mechanism presented in chapter 4.

Hardware abstraction (Task 13) as the last requirement is complied with the method proposed in agent-based manufacturing. As assessed in the literature review in section 2.4, an encapsulated Resource Layer, adapted from approaches such as PROSA [Van Brussel et al., 1998] or ADACOR [Leitão and Restivo, 2008], is proposed for single manufacturing components in this thesis and used as the hardware abstraction layer of the mobile commissioning robot. The disadvantage of hardware abstraction is the lack of low level knowledge which can be mitigated in future work by combining MobComm with a mechanism that allows to react to hardware failures and production flow changes such as proposed in ADACOR [Leitão and Restivo, 2008] (cf. figure 2.19 on page 43).

The proposed MobComm architecture, that implements the holonic principle in standard execution and reconfiguration activities, focuses on a self-organising Skill Layer with a dynamic service allocation in runtime. These characteristics qualify this architecture as a basis for the reconfiguration mechanism introduced in the following chapter.

Chapter 4

Design of MobComm Reconfiguration Mechanism

Now that the characteristics of MobComm architecture have been explained in chapter 3, the self-organised reconfiguration mechanism is detailed in the following. The proposed mechanism results in a provision of missing robot functionalities. As an introduction, the overview of the reconfiguration activities is given in figure 4.1.

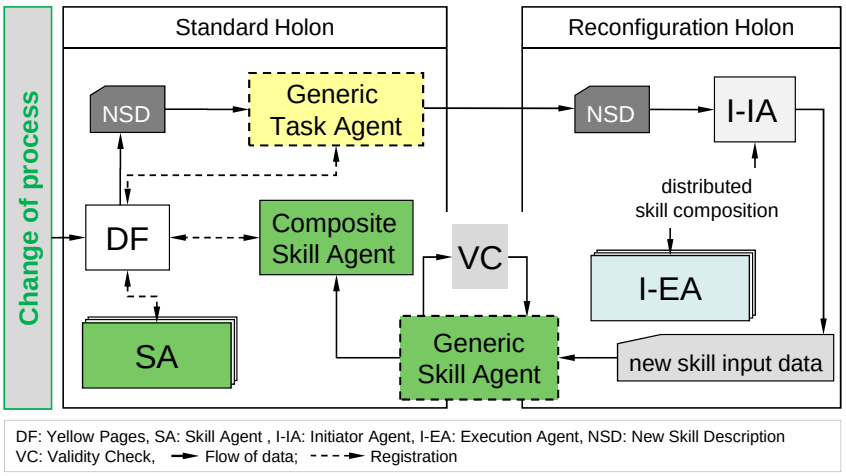


Figure 4.1: Overview of reconfiguration activities from the functional process change to the insertion of a Composite Skill Agent in Standard Holon.

The activation of a reconfiguration process implies the insertion of a functional process change by the operator. This process change is encapsulated in a data structure called New Skill Description (NSD) which is used to communicate new process requirements to

the Reconfiguration Holon. The missing functionality that is included in the New Skill Description triggers the initialisation of a Reconfiguration Holon.

The agents contained in Reconfiguration Holon are designed and implemented as BDI-agents with the goal to create a new Composite Skill Agent during the mechanism. The agent that results from a MobComm reconfiguration is thereafter integrated in the standard process execution by making its new service available to Standard Holon. As discussed in the architecture chapter 3, a Reconfiguration Holon is ordered as a heterarchy with entities that have higher autonomy than in Standard Holon and additionally contain a world model. BDI-agents are further able to handle knowledge-intensive reconfiguration tasks with the indication as interaction agents "I-*AgentName*" in this thesis.

The chronological order of this chapter follows the flow of reconfiguration as given in figure 4.2. Three sections outline the total reconfiguration process in the following. While starting with the creation of the Reconfiguration Holon in section 4.1, the distributed skill composition is further elaborated in section 4.2. Presuming a successful composition, section 4.3 describes the generation of the new agent, before the self-organising properties are discussed in section 4.4.

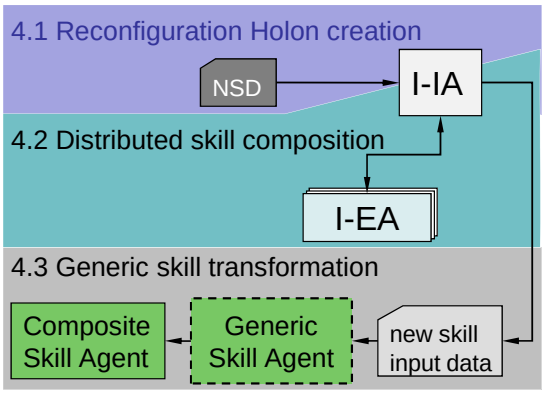


Figure 4.2: Structure of Reconfiguration Mechanism Chapter.

4.1 Creation of Reconfiguration Holon

The creation of a Reconfiguration Holon following the insertion of a functional process change is described in this section. Compared to the behaviour-based and hierarchical-organised agents used in Standard Holon as presented in figure 3.5 on page 80, the self-organised reconfiguration mechanism in Reconfiguration Holon requires a flexible and dynamic interaction mechanisms with goal-driven planning capabilities as outlined in the

following.

4.1.1 Agent Types and Interaction

The set of agents used in Standard Holon provides predefined functionalities in encapsulated behaviours to handle messages protocol-specific, while the pre-coded behaviours can be activated and scheduled dynamically. The reconfiguration mechanism, however, requires a self-organising communication structure with goal-driven agents. With reference to the review of agent concepts in section 2.3, the BDI paradigm is a widespread way to model cognitive agents with the provision of a goal-directed view [Bellifemine et al., 2007]. BDI agents have the capability to individually interpret the content of messages depending on the state of their belief base.

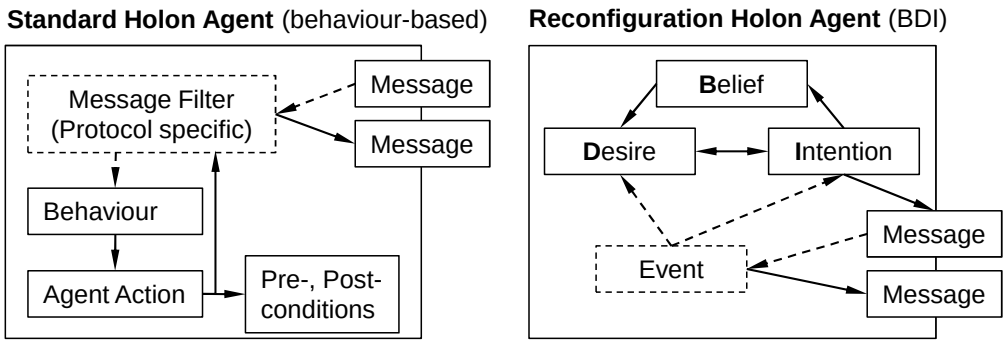
Behaviour-based agents, as used in Standard Holon and presented in figure 4.3(a), are not able to react dynamically to changes in the environment but provide an effective mechanism to execute agent actions triggered by a filtered message. Message filters and agent behaviours are mapped in the Preconditions and Postconditions of the agent.

In general, a Precondition must hold true for an agent "to be able to perform the action" [Bellifemine et al., 2007] and thus requires the compliance with the set of Preconditions before a skill can be executed. In MobComm the set of Preconditions is composed of activation messages, resource allocation, and ontology variables.

The Postconditions, however, represent "the effect that the agent considers to be true just after the execution of the action" [Bellifemine et al., 2007]. Accordingly, agent outcome, resource allocation, and ontology variables are contained in the set of Postconditions of agents used in Standard Holon.

In contrast to behaviours triggered in Standard Holon-agents, the messages received by the agents in Reconfiguration Holon cause an internal event as shown in figure 4.3(b). Events in turn are able to modify agent desires and intentions. The desires of a BDI agent can be modified externally through messages and by the own belief base. While desires and intentions are influenced by each other, beliefs can be modified by varying intentions of an agent. These modifications can affect an agent internally, as well as other instances of the same agent type, and different types of BDI agents with the result of dynamic reactions to changing aims within the Reconfiguration Holon.

For the execution of the reconfiguration mechanism, three different types of cognitive agents are provided: The Initiator Agent (I-IA) contains the management and decision



(a) Agent interaction in Standard Holon. (b) Agent interaction in Reconfiguration Holon.

Figure 4.3: Comparison of agent behaviour in Standard and Reconfiguration Holon.

functionalities, while the set of Execution Agents (I-EA) is responsible for the composition of existing functionalities. The Validator Agent (I-VA) executes the Validity Check following a successful composition. Figure 4.4 outlines the interaction in Reconfiguration Holon including the New Skill Description (NSD) structure that initialises the reconfiguration and the New Skill Input Data (NSID) that maps the data structure required for the transformation in a Generic Skill Agent (GSA).

The main task of the Reconfiguration Holon is the interaction between the Execution Agents (I-EA) that are individually linked to one Cloned Skill Agent (Cl-SA). Agent cloning and its usage in MobComm is further outlined in section 4.1.2. Besides the initialisation of reconfiguration, its finalisation is also concentrated in the Interaction Agent (I-IA) as well.

Even if the core part of the interaction in the Reconfiguration Holon is heterarchical, a hierarchy is still required between the Initiator and the Validator Agent for the activation of the Validator Agent dependent on the preceding reconfiguration results. These results, generated jointly by the set of Execution Agents, are evaluated by the Initiator Agent (I-IA) and validated by the Validator Agent in case of a positive outcome.

BDI agents used in the Reconfiguration Holon are characterised by its beliefs, desires, and intentions as detailed in table 4.1. The Initiator Agent, as the first agent created, configures the agents in the Reconfiguration Holon for the new reconfiguration process. The set of Execution Agents is created in line with the requirements of the New Skill Description. Both initialisation and finalisation of the reconfiguration mechanism are handled by the I-IA including collection of the results and activation of the Validator Agent. In order to comply with the named desires, the according beliefs are stored in the

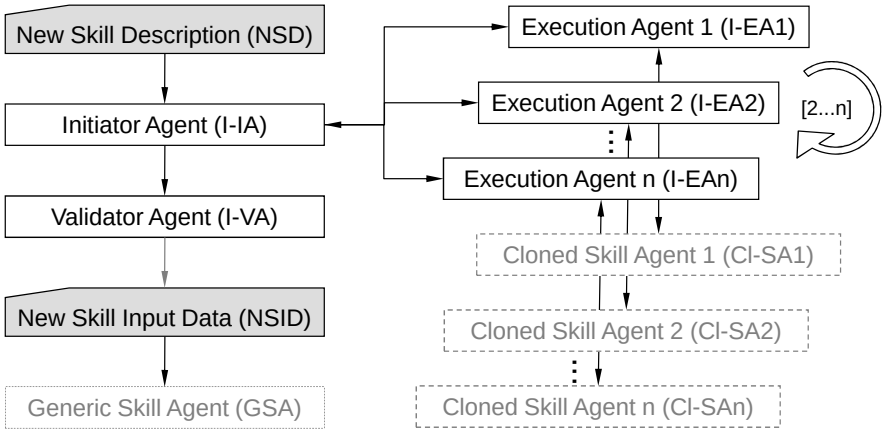


Figure 4.4: Overview of interaction in the Reconfiguration Holon.

belief base of the Initiator Agent and intentions are pre-programmed as concrete executable plans as given in table 4.1. For the execution of the reconfiguration mechanism state of the art planning methods are used as provided by the BDI framework Jadex [Braubach et al., 2004]. A survey of planning and reasoning including the BDI principle is given in [Clement et al., 2007].

Initiated by the plans of the Initiator Agent, the set of Execution Agents desires the creation of the new robot functionality jointly with the result of a Composite Skill Agent. Every Execution Agent has integrated beliefs that consist of the distributed Standard Holon-knowledge as further described in the next section 4.1.2. In case the Validator Agent (I-VA) is started by the Initiator Agent (I-IA), its desire to execute the Validity Check is activated prior to the integration of the Composite Skill Agent in Standard Holon.

	Beliefs (Data structures)	Desires (Abstract goals)	Intentions (Concrete plans)
Initiator Agent I-IA	NSD components; Reconfiguration parameters; Matching reports.	Initiation of reconfiguration; Collection of matching reports.	Plan to prepare RH for a successful reconfiguration; Plan to collect reconfiguration results.
Execution Agent I-EA	Cloned SH-knowledge.	Creation of a new Composite Skill Agent.	Plan to build new Skill Agent with other I-EAs.
Validator Agent I-VA	NSID; VC parameters; GSA parameters.	Execution of Validity Check; Integration of new agent in Standard Holon.	Plan to execute VC; Plan to integrate the new agent in Standard Holon.

Table 4.1: BDI aspects of reconfiguration agents.

Based on the BDI capabilities of reconfiguration agents, the integration of knowledge

used during standard execution is the next step towards a self-organised generation of the required robot functionalities.

4.1.2 Integration of Standard Holon Knowledge

The integration of Standard Holon knowledge contains the semantic exploitation of the New Skill Description (NSD) and the use of Skill Layer knowledge extracted by agent cloning.

The New Skill Description (NSD) contains semantic information about the functional process change and the required new robot functionality. This data structure is detailed in figure 4.5 and also described as a concept in the MobComm ontology, given in figure 3.11 on page 91. The New Skill Description that is created out of the operator's semantic description is not a consistent data structure for the use in the reconfiguration process but rather states the semantic requirements of the operator concerning the new functionality.

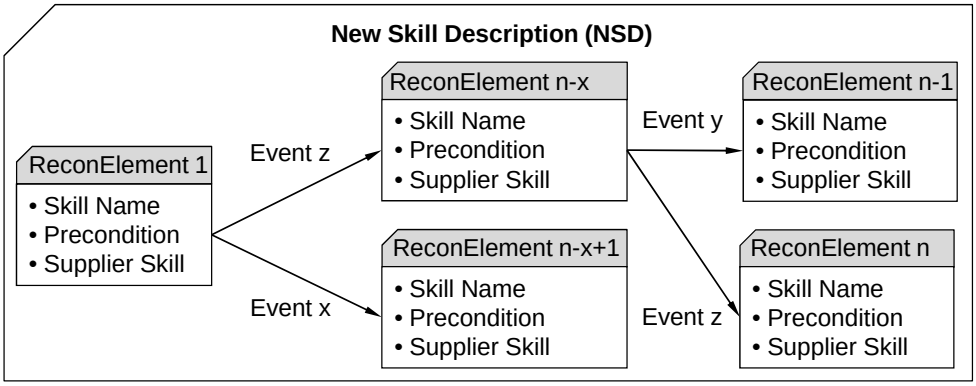


Figure 4.5: Structure of the New Skill Description (NSD) concept.

The core part of the New Skill Description is the set of Reconfiguration Elements (abbreviated ReconElements in figure 4.5) and the connecting *Events* that can both be extracted from the New Skill Description. Reconfiguration Elements display atomic components of the new robot functionality while they are connected to the surrounding components by *Events*. The actual structure of MobComm *Events* covers conditions like loops, if-else-clauses, or breaks. The optional variable *Supplier Skill* provides the suggestion of the operator for a possible predecessor of the used skill.

The *Events* integrated in the New Skill Description outline the scheduling distribution as evaluated in section 3.3.1. Temporal scheduling is contained in Task Layer and thus not required for the skill-based mechanism in Reconfiguration Holon. This is in contrast

to conditional aspects that are contained in the used Reconfiguration Elements. The conditional range of *Events* is basically not limited in terms of complexity and used variables as long as the according input options are provided by the user interface and mapped into the MobComm ontology. The presented work implements a restriction to *Events* as only hardware variables and existing ontology concepts are permitted as their input. The integration of new ontology concepts is directed to future work as well as the generation of complex conditional path structures in the *Events*.

Even if the New Skill Description is based on manually-inserted semantics, its consistency and complexity constitute high influence on the composition mechanism as introduced in section 4.2.

Besides the exploitation of the New Skill Description, the integration of Skill Layer knowledge is applied for an encapsulated and independent reconfiguration mechanism.

The knowledge about agent behaviours and characteristics in Standard Holon is provided decentralised by the set of Preconditions and Postconditions attached to every Standard Holon-agent (cf. figure 4.3(a)). These conditions provide the basis for the knowledge access in Reconfiguration Holon by cloning the used Standard Holon-agents. The use of agent cloning emphasises the self-organising principle as described in [Shehory et al., 1998]. Agent cloning is the process by which an agent replicates itself as found in biological system evolution and is adapted by agent platforms as JADE [Bellifemine et al., 2007] or Jadex [Pokahr et al., 2005].

The application of agent cloning is restricted to Skill Layer in MobComm whereas every Execution Agent initialises the cloning of a specific Skill Agent as introduced in figure 4.4. Thus, the set of Execution Agents accesses the behaviour descriptions of Standard Holon-agents decentralised by the Preconditions and Postconditions of Cloned Skill Agents (Cl-SA). The agents' behaviour characteristics are integrated into the belief bases of the Execution Agents as shown in figure 4.6.

The integration of Standard Holon knowledge in the cognitive agents in Reconfiguration Holon allows the implementation of self-organisation (Research Task 1) due to the cognitive and goal-oriented agent specifications. The dynamic integration of Standard Holon-knowledge into the belief bases of Execution Agents permits the realisation of robot configuration independence and hardware abstraction (cf. Task 4/5). Regardless of whether the configuration in Standard Holon or the robot hardware changes, the reconfiguration mechanism is still applicable as only the agent plans are pre-coded while beliefs and agent

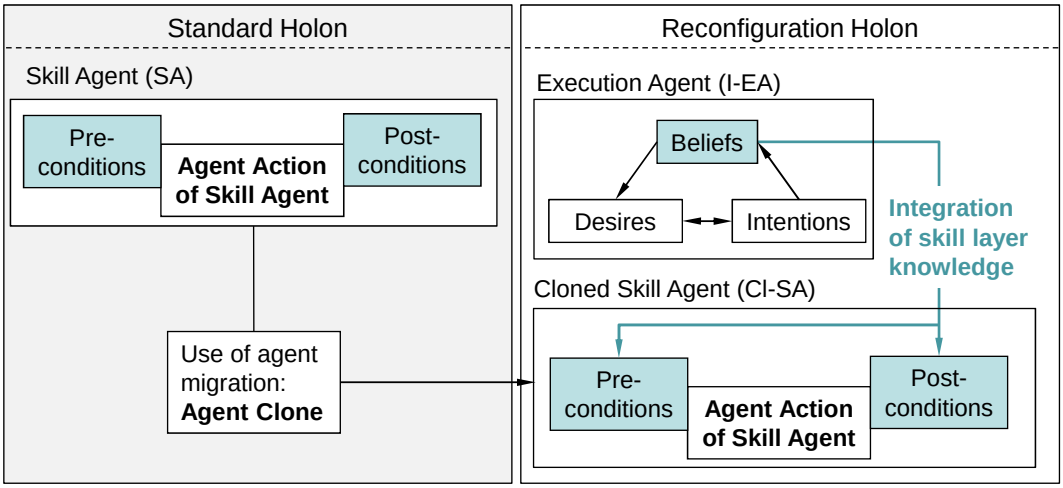


Figure 4.6: Integration of Standard Holon-knowledge into the reconfiguration mechanism by agent cloning.

goals are loaded dynamically during reconfiguration.

For the compliance of the required maintenance of productivity (Research Task 2), MobComm architecture provides the basis with its abstraction of hardware into Resource Layer. The compliance is further enhanced in the Reconfiguration Holon by the application of agent cloning. By the execution of agent cloning, a MobComm reconfiguration is completely decoupled from the running process in Standard Holon and consequently does not affect its productivity. The described context between the reconfiguration characteristics and the according task compliances are summarised in figure 4.7.

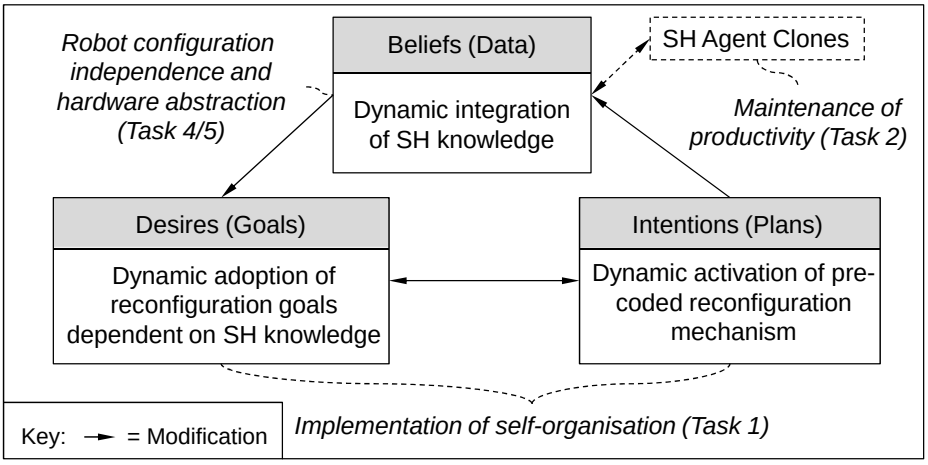


Figure 4.7: BDI aspects of reconfiguration agents referring to the compliance of given research tasks.

By using Standard Holon knowledge, the set of Execution Agents is able to dynamically

recombine given robot skills independent from configurations or hardware specifications as introduced in the following.

4.2 Distributed Skill Composition

While the last section described the function principles of the Reconfiguration Holon and the integration of Standard Holon-knowledge into the cognitive agent structure, this section focuses on the execution of the distributed skill composition following figure 4.2 on page 95. To get an overview over the distributed skill composition, an example communication sequence is presented in figure 4.8 with a division into three parts.

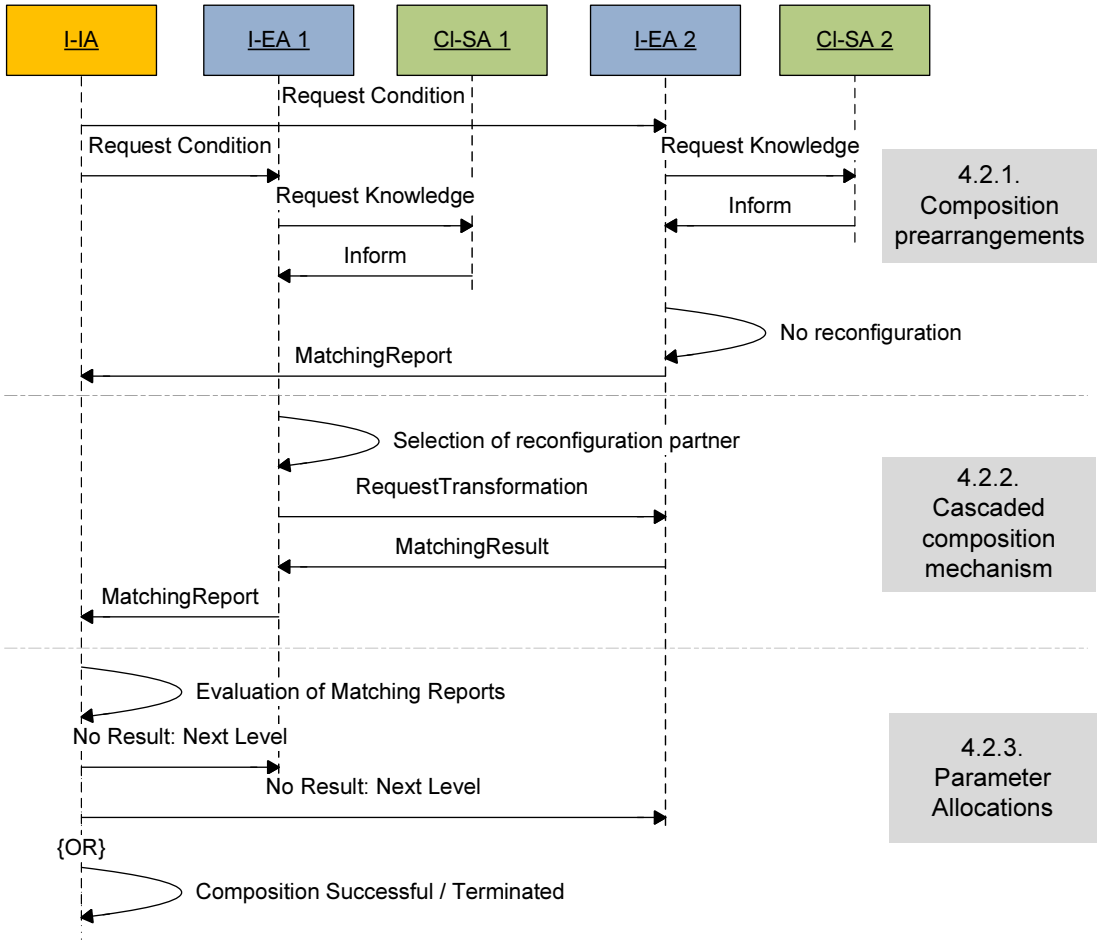


Figure 4.8: Example communication structure during the distributed composition mechanism with its division into three parts.

The composition prearrangements are explained in section 4.2.1. While the cascaded composition mechanism is introduced in section 4.2.2, the resulting parameter allocations

are detailed in section 4.2.3. Before these steps are described, an overview of the core reconfiguration components and their entity-relationship diagram are given in the following.

As described in table 4.2, the basic concepts for skill composition are the New Skill Description and its contained Reconfiguration Elements as a semantic description of the functional process change. The Reconfiguration Elements reflect the encapsulated skills used in the New Skill Description. Every Reconfiguration Element of the New Skill Description is linked to an Execution Agent that is connected in turn to a Cloned Skill Agent. This association is required for the integration of Standard Holon-knowledge as introduced in section 4.1.2.

Name	Type	Description
New Skill Description (NSD)	Data structure	Semantic description of operator's requirements for new functionality.
Reconfiguration Element (ReconElement)	Data structure (part of NSD)	Data structure containing the decomposed NSD i.e. Used Skill, Supplier Skill, Precondition, Event.
Execution Agent (I-EA)	BDI agent	An I-EA is initiated for every Reconfiguration Element. The used skill is cloned from SH and linked to the I-EA.
Request Condition	Message	A Request Condition is sent from the I-IA to the I-EAs to initiate the skill composition. Its answer is a Matching Report.
Request Transformation	Message	Request Transformation is sent among I-EAs for the processing of the Request Condition.
Matching Report	Data structure	Matching Reports are sent from I-EAs to I-IA following the execution of a Request Condition.

Table 4.2: Components used for the distributed skill composition.

As outlined in table 4.2, two novel message contents are introduced to facilitate the execution of skill composition. The *RequestCondition*-message is sent from the Initiator Agent to any Execution Agent for the initiation of the composition. The *RequestTransformation*-message, however, is sent among Execution Agents following the identification of reconfiguration needs. The execution of the transformation associated with this message is detailed in section 4.2.2.

Following the entity-relationship diagram in figure 4.9, the number of *RequestCondition*- and *RequestTransformation*-messages is dependent on the applied composition level as introduced in section 4.2.2. Every *RequestCondition*-Message must be answered by the addressed Execution Agent with a *Matching Report* that contains the local composition

results.

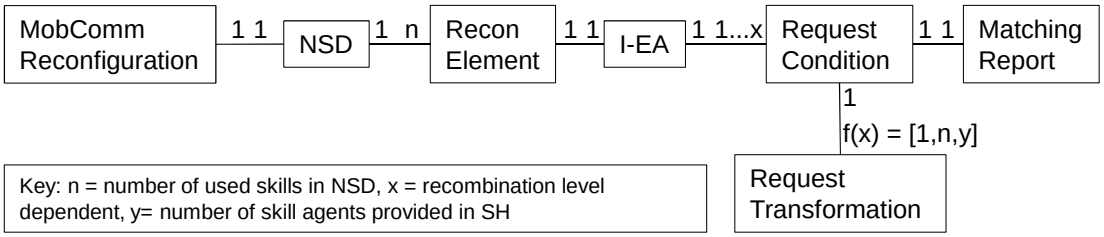


Figure 4.9: Entity-relationship diagram of MobComm reconfiguration.

After the introduction of the main composition components and their relations, the prearrangements are described in the next section following figure 4.8.

4.2.1 Composition Prearrangements

The prearrangements required for the execution of the skill composition are embedded in the *Prepare for reconfiguration*-plan of the Initiator Agent as presented in table 4.1 on page 98. To initialise the skill composition, the New Skill Description (NSD) is decomposed into its Reconfiguration Elements before the same number of Execution Agents is activated by the Initiator Agent in the first composition level. The resulting set of Execution Agents subsequently initiates the agent cloning and the associated integration of Standard Holon-knowledge as described in figure 4.4.

Once the Standard Holon-knowledge is integrated and skill composition is initialised by a *RequestCondition*-Message of the Initiator Agent, this message is analysed separately in every Execution Agent to establish the local requirements of skill composition. Every Execution Agent checks whether the desired Precondition in its Reconfiguration Element matches the Precondition provided by its attached Cloned Skill Agent. Figure 4.10 presents a condition check by using the example of SA_{move}. In the given example, the *Position*-Precondition of the Reconfiguration Element mismatches with the *Location*-Precondition of SA_{move}. This mismatch entails the requirement of the composition mechanism as described in the next section.

Should the required and the provided Preconditions match within an Execution Agent, a Matching Report is sent to the Initiator Agent with the content that no reconfiguration is required. The condition analysis of the Execution Agents concerning the requirement of composition concludes the prearrangements of the mechanism and initiates the distinct composition levels in the next section.

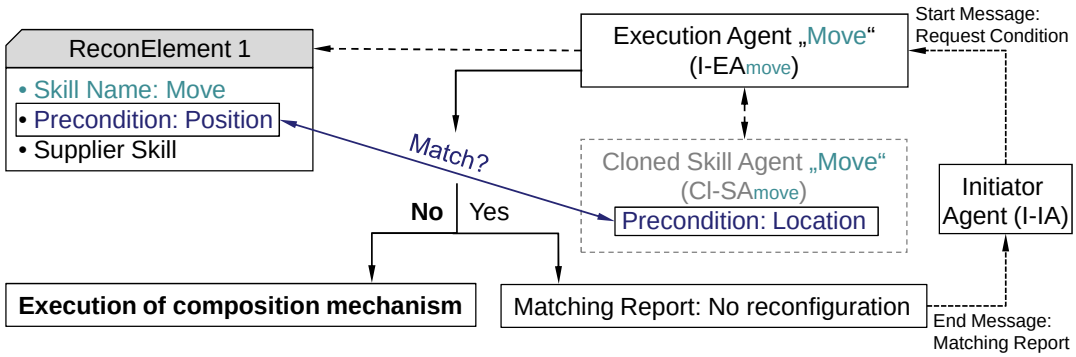


Figure 4.10: Example condition matching using the example of skill SA_{move}.

4.2.2 Cascaded Composition Mechanism

Following figure 4.8, the execution of the cascaded composition mechanism succeeds the composition prearrangements. The mechanism is initiated by an Execution Agent that identified a mismatch between a Required and a Provided Precondition as presented in figure 4.10. The temporal sequence of the composition execution is negligible as the skill sequence is later sorted by the Initiator Agent as explained in section 4.2.3. The composition mechanism is based on the application of a distributed backwards search mechanism and is dedicated to realise a new skill sequence without condition mismatches.

An Execution Agent that identifies a mismatching pair of conditions requires an Execution Agent or a sequence of them as a search result to overcome the imbalance of conditions. Figure 4.11 overviews the desired search result for the example Execution Agent $I-EA_{move}$ to solve the mismatch between the *Location*- and the *Position*-Precondition.

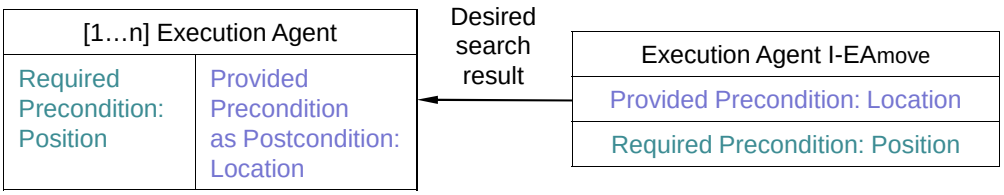


Figure 4.11: Required search result for Execution Agent $I-EA_{move}$.

To elucidate the composition components introduced in table 4.2, an example New Skill Description $NSD_{gripAny}$ is given in the following to demonstrate the required steps for a successful skill composition. The operator desires to "grip a detected *EnvObject* X at the detected *Location*". An *EnvObject* is an ontology concept that maps an object provided in the application of the robot system as introduced in figure 3.11. $NSD_{gripAny}$ is decomposed into two Reconfiguration Elements whereas connecting *Events* are not regarded in

this example:

- Reconfiguration Element 1: Skill Name: *Detect*; Precondition: *EnvObject*.
- Reconfiguration Element 2:

  Skill Name: *Manipulate*; Precondition: *Location*; Supplier: *Detect*.

These Reconfiguration Elements are linked to the Execution Agents $I-EA_{detect}$ and $I–EA_{manipulate}$ due to their used skills in the composition prearrangements as described in section 4.2.1. Additionally the Cloned Skill Agents $Cl-SA_{detect}$ and $Cl-SA_{manipulate}$ are created and attached to the Execution Agents. Thereafter, $I–EA_{detect}$ and $I–EA_{manipulate}$ check internally if the *RequestCondition* sent by the Initiator Agent has to be handled or if a Matching Report is directly sent back as described in figure 4.10.

For a better comprehension of the search mechanism throughout this section, table 4.3 provides an extract of the Preconditions and Postconditions of Atomic Skill Agents implemented in the mobile commissioning robot. The information, overviewed in table 4.3, is distributed in MobComm to maintain the independence of robot configuration for the reconfiguration mechanism.

BDI agent	Related Skill Agent	AgentAction	Precondition	Postcondition
I-EA$_{moveArm}$	SA$_{moveArm}$	MoveArm	Position	Position
I-EA$_{movePlfm}$	SA$_{movePltf}$	MovePltfToLoc	Location	Location
		MovePltfToPos	Location	Position
I-EA$_{detect}$	SA$_{detect}$	DetectLocation	EnvObject	EnvObject; Location
		DetectPosition	EnvObject	EnvObject; Position
I-EA$_{manipulate}$	SA$_{manipulate}$	Grip	Position	EnvObject; Position
		Deposit	EnvObject; Position	Position

Table 4.3: Preconditions and Postconditions of Atomic Skill Agents.

Following table 4.3 in the presented example, $I-EA_{detect}$ is able to comply with the Precondition *EnvObject* as desired in *Reconfiguration Element 1*. Accordingly, $I–EA_{detect}$ sends a Matching Report to the Initiator Agent and requires no further actions in this example composition.

The Execution Agent $I–EA_{manipulate}$ has to handle a *Location*-Precondition of *Reconfiguration Element 2* whereas it can only provide the *Position*-Precondition as an input

variable for *AgentAction Grip* and an *EnvObject* to *deposit* an *EnvObject*. Thus a skill composition must be initiated by $I-EA_{manipulate}$.

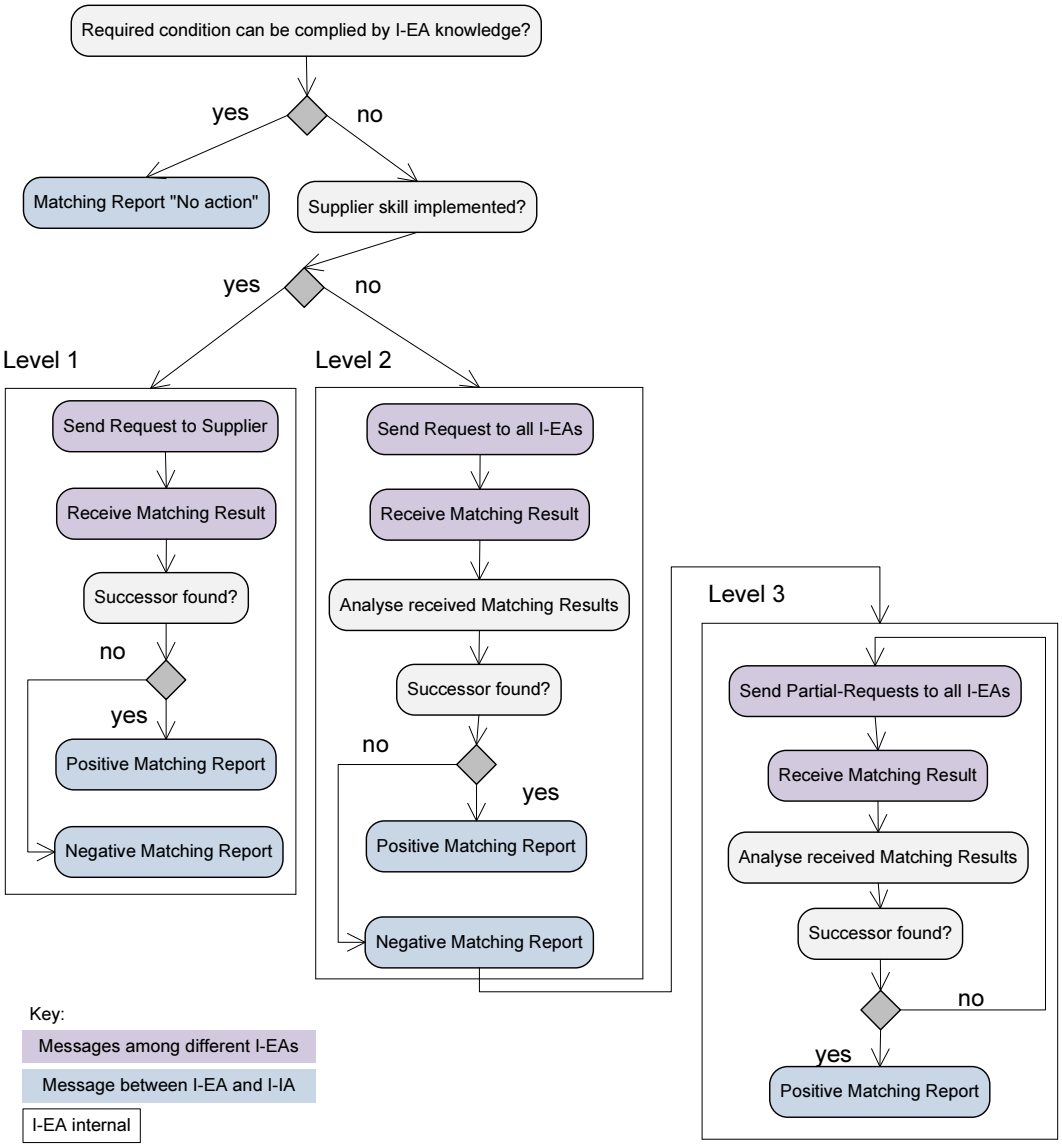


Figure 4.12: Sequence diagram of the composition levels including their interfaces to other BDI agents.

The presented skill composition provides three independent composition levels according to the consistency of the entry data used in the New Skill Description. In the first composition level, the Initiator Agent creates only the number of Execution Agents appropriate to the number of Reconfiguration Elements in the NSD. Thus, in the example of $NSD_{gripAny}$, $I-EA_{manipulate}$ is linked to *Reconfiguration Element 2* and has the Skill Agent SA_{detect} as an implemented supplier skill. Thus, $I-EA_{manipulate}$ sends a *Re-*

questTransformation to $I-EA_{detect}$ with the content of a *Location*-Precondition and a *Position*-Postcondition. As $I-EA_{detect}$ cannot comply with the desired conditions following table 4.3, a negative *Inform*-message is sent back to $I-EA_{manipulate}$ which activates the second composition level.

The second and third levels of composition allow to handle a declining compatibility of used skills in the New Skill Description, whereas the first level only succeeds if a high compatibility within the implemented skills is given. The second level, however, is no longer dependent on a provided supplier skill in the NSD. All Skill Agents integrated in Skill Layer are cloned and linked to an Execution Agent in Reconfiguration Holon.

According to this process, the Execution Agent $I-EA_{manipulate}$ requests a *Location*-Precondition and a *Position*-Postcondition from the complete set of Execution Agents. In our example, only $I-EA_{move}$ is able to fulfil the request. A positive Matching Report is returned to I-IA and the composition mechanism can be terminated with the second level following the sequence diagram in figure 4.12. The sequence of skills for $NSD_{gripAny}$ results as SA_{detect}, SA_{move}, $SA_{manipulate}$.

Should it not be possible to match any of the conditions in the second composition level, the activation of level three as described in figure 4.12 is initiated. Even if the provided Reconfiguration Elements are formally not changed as presented in figure 4.13, a higher incompatibility of used skills can be balanced in the third level.

A *Partial-Condition Request* is sent to the set of available Execution Agents where an allocation of either the Precondition or the Postcondition already permits the transmission of a positive Matching Result by the requested Execution Agent. In case a *RequestTransformation* that is sent to all Execution Agents contains a *Position*-Precondition and a *Location*-Postcondition, the Execution Agents $I-EA_{moveArm}$ and $I-EA_{manipulate}$ send a Partial-Agree as they are able to comply with the *Position*-Precondition according to table 4.3. $I-EA_{movePltf}$ and $I-EA_{detect}$ on the other hand are able to give a Partial-Agree to the *Location*-Postcondition.

The set of Partial-Agrees results in *RequestTransformation*s with adapted condition requirements. Emerging from the Partial-Agree of $I-EA_{movePltf}$ and $I-EA_{detect}$ the following *RequestTransformations* are sent to all Execution Agents:

- (Sender: $I-EA_{movePlfm}$ (RequestTransformation
  (Required Precondition: *Position*; Required Postcondition: *Location*)))

- (Sender: $I-EA_{detect}$ (RequestTransformation
  (Required Precondition: *Position*; Required Postcondition: *EnvObject*)))

The Request of $I-EA_{movePlfm}$ cannot be complied by any Execution Agent as none can offer a *Position*-Precondition and a *Location*-Postcondition, in contrast to the Request of $I-EA_{detect}$ that is corresponded by $I-SA_{manipulate}$ with the required *Position*-Precondition and *EnvObject*-Postcondition. Thus, the skill combination results as $SA_{manipulate}$, SA_{detect}, and SA_{move}.

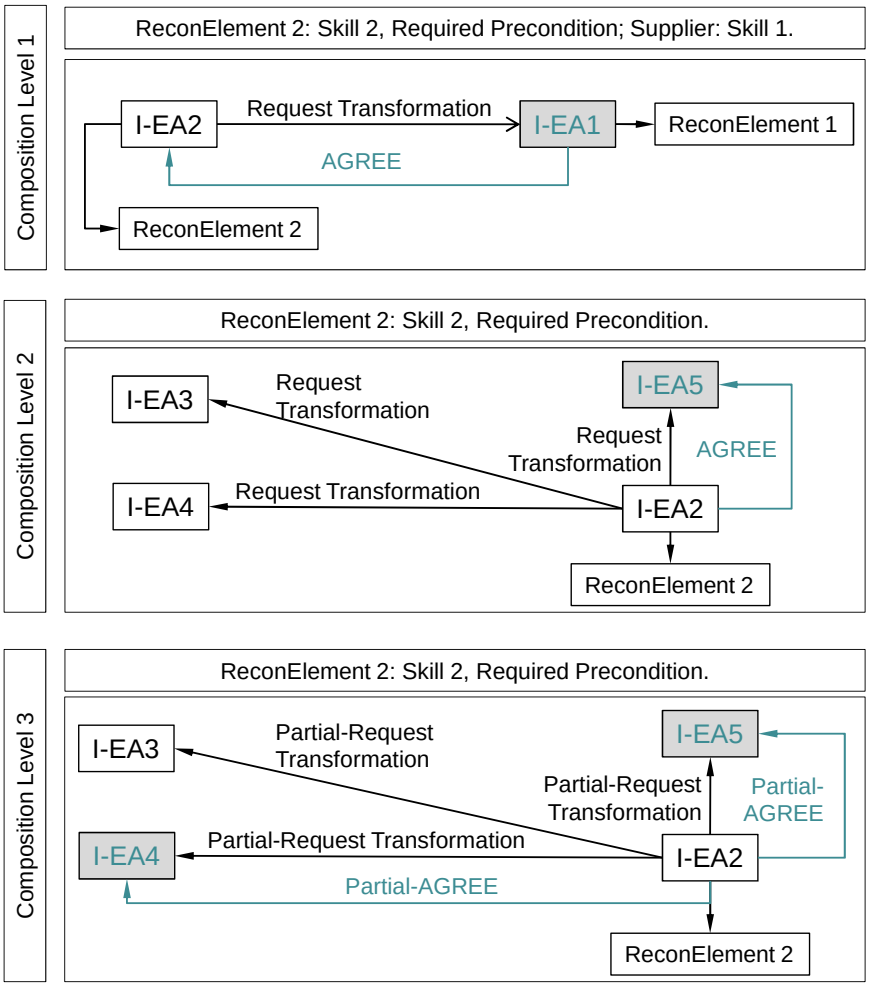


Figure 4.13: Overview of composition levels.

All composition levels conclude with the provision of a Matching Report containing the local composition results. These results are analysed by the Initiator Agent as described in the next section.

4.2.3 Reconfiguration knowledge extraction

Following the execution of the composition mechanism as described in the previous section, the set of *Matching Reports* is collected by the Initiator Agent and further processed. During this analysis, the goal of the Initiator Agent is the creation of parameter allocations and a resource schema. While parameter allocations map the structure of the resulting agent following the integration of the set of *Events*, the resource schema is required for the Validity Check introduced in chapter 5.

A Matching Report contains a set of parameters as presented in figure 4.14. By providing both the preceding Agent and the attached Interaction Agent, a set of parameter allocations can be built of the *Matching Reports*. For the preparation of the parameter allocations, the content of the single *Matching Reports* is saved in the belief base of the Initiator Agent.

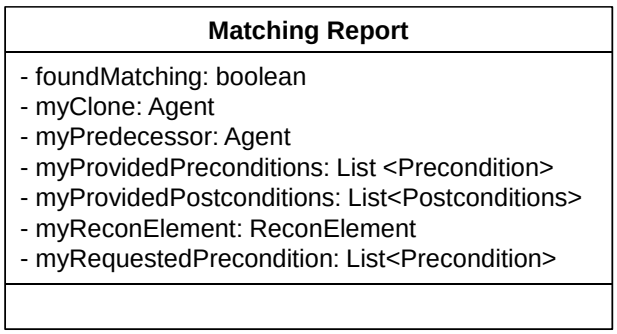


Figure 4.14: Structure and content of a Matching Report.

As soon as the set of Matching Reports is complete and stored in the belief base, the Initiator Agent executes the parameter allocation algorithm as described in figure 4.15. The starting point is the sorting of the *Matching Reports* according to the Index of the Reconfiguration Elements. This process establishes the basic skill skeleton following the inserted New Skill Description. A refinement of the Matching Report sequence is performed by the analysis of the *myPredecessor* parameter in the *Matching Reports*. Once the *Matching Reports* are provided already sorted, the single Events contained in the Reconfiguration Elements are linked to the skills of the sorted *Matching Reports*. The resulting list of subsequent skills as presented in figure 4.15 is thus dependent on the compliance of the related Events.

Once the set of Preconditions and Postconditions has been set to the indexed skills, the desired list of parameter allocations can be created based on the developed data structure

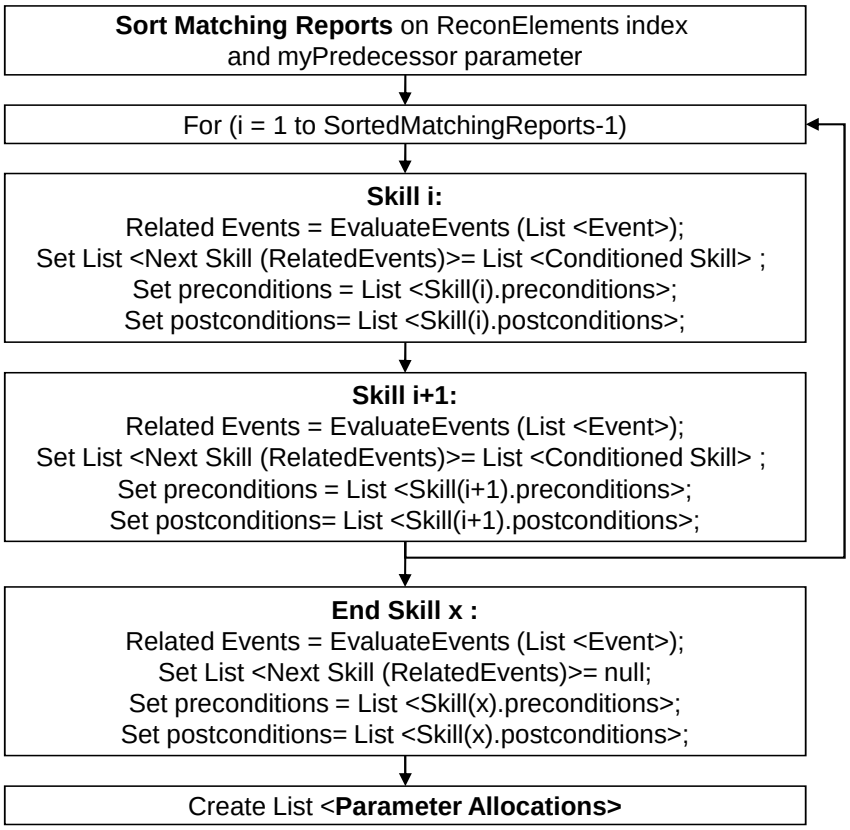


Figure 4.15: Algorithm to generate parameter allocations by the Initiator Agent.

in the belief base of the Initiator Agent. As illustrated in figure 4.16, the parameter allocations are comprised of the possible combinations between the Postconditions of a Skill and the Preconditions of the list of succeeding skills (cf. $List < NextSkill(RelatedEvents) >$ in figure 4.15). To prepare the parameter allocations for its transformation into a Generic Skill Agent in the next section, the allocations are verified regarding their completeness and content-related correctness.

If inconsistencies in the resulting parameter allocations are analysed by the Initiator Agent, the MobComm reconfiguration is cancelled completely. By the introduction of an error classification and the implementation of reaction strategies as agent plans, alternative composition partners can be searched by the affected Execution Agent in future work.

In contrast to the evaluation of the parameter allocations during the reconfiguration mechanism, the resource schema is generated for the execution of the Validity Check as detailed in chapter 5. Only the extraction of reconfiguration knowledge during the reconfiguration mechanism allows a succeeding validation of the reconfiguration results in real-world.

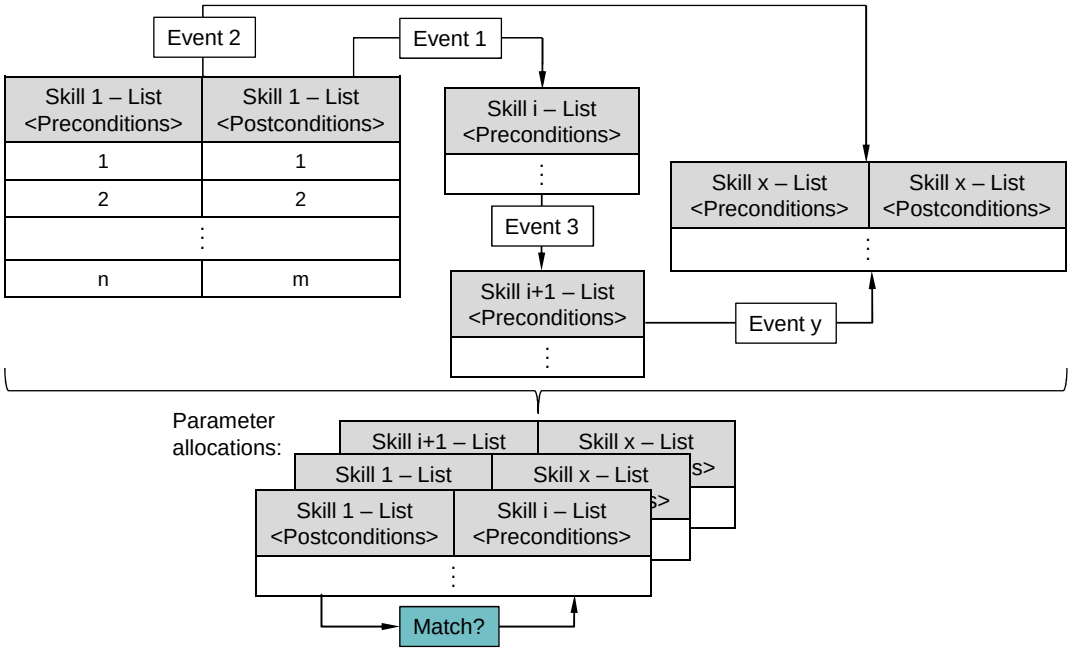


Figure 4.16: Generation of lists of parameter allocations for a new robot functionality.

The resource schema is an extracted data set of the parameter allocations. All Preconditions *Resource Service* are extracted from the used skills including their sequence. Thus, a data set occurs that contains the used Cloned Skill Agents and the according Resource services as illustrated in figure 4.17.

After a successful verification of parameter allocations and the generation of the resource schema, the Generic Skill Transformations as described in the next section are initialised by the Initiator Agent.

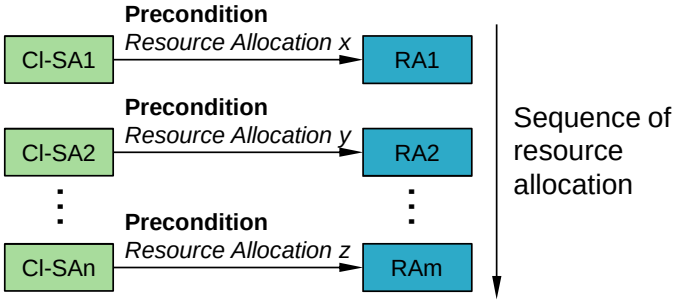


Figure 4.17: Resource schema for the provision of the Validity Check.

4.3 Generic Skill Transformation

Following the preparation of the parameter allocations, the finalising step in the composition mechanism are the Generic Skill Transformations. Even if the transformations are proceeded in Reconfiguration Holon by BDI agents, the resulting Skill Agent complies with interaction and communication standards of behaviour-based Skill Agents as used in Standard Holon and introduced in figure 4.3(a) on page 97. Figure 4.18 overviews the individual transformation steps until the new Generic Skill Agent can be integrated into Standard Holon as an example communication sequence diagram.

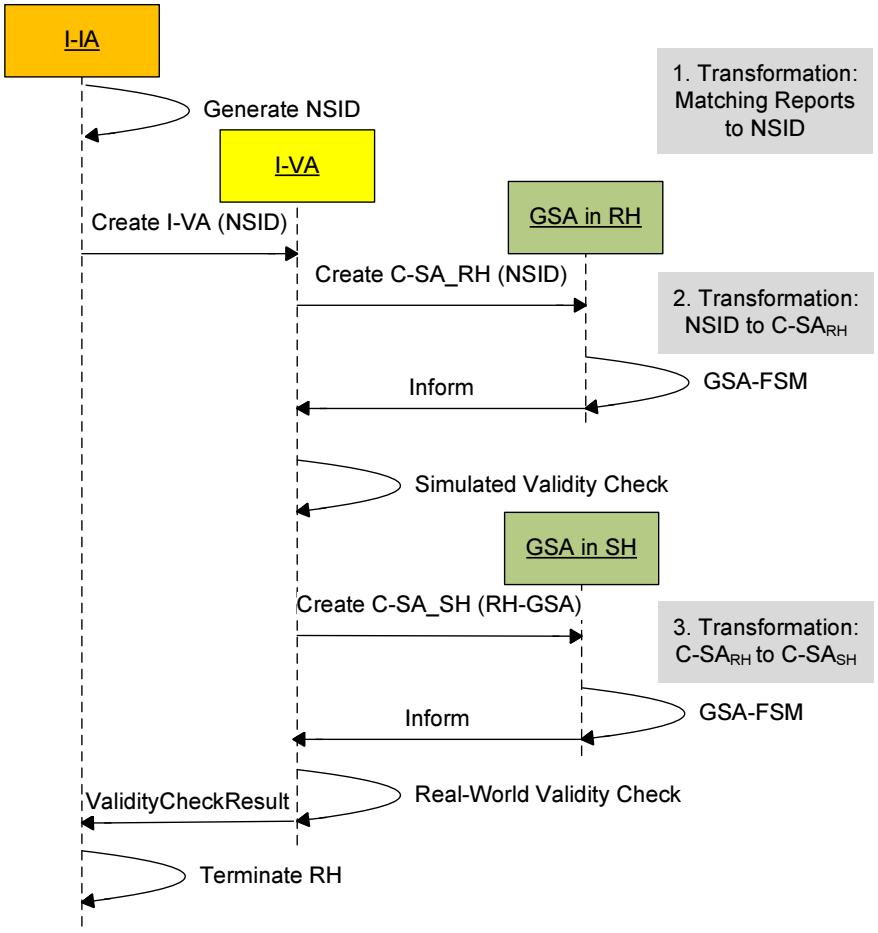


Figure 4.18: Overview of Generic Skill Transformations by an example communication sequence.

Three different transformations are required that include the transformation from the parameter allocations to the New Skill Input Data (NSID) executed by the I-IA (cf. first transformation in figure 4.18), further from the NSID to the Composite Skill Agent $C-SA_{RH}$ (cf. second transformation in figure 4.18), and finally the Composite Skill Agent

in Standard Holon $C-SA_{SH}$ (cf. third transformation in figure 4.18) as a plan of the Validator Agent.

The transformation from the parameter allocations to the New Skill Input Data (NSID) structure prepares the conversion from data about the new skill into a Finite State Machine (FSM) that is able to formalise and preserve the provided knowledge of the Composite Skill Agent. The parameter allocations are transferred into the input and output variables of the single *FSMDescriptions* while one *FMSDescription* represents a *state* in the resulting *Finite State Machine* as presented in figure 4.19. The NSID is still a data structure that is kept locally in the belief base of the Initiator Agent. The initiation of the Validator Agent and the transfer of the NSID is the terminating action of the Initiator Agent while the remaining transformations are executed by the Validator Agent as they are integrated in the Validity Check detailed in chapter 5.

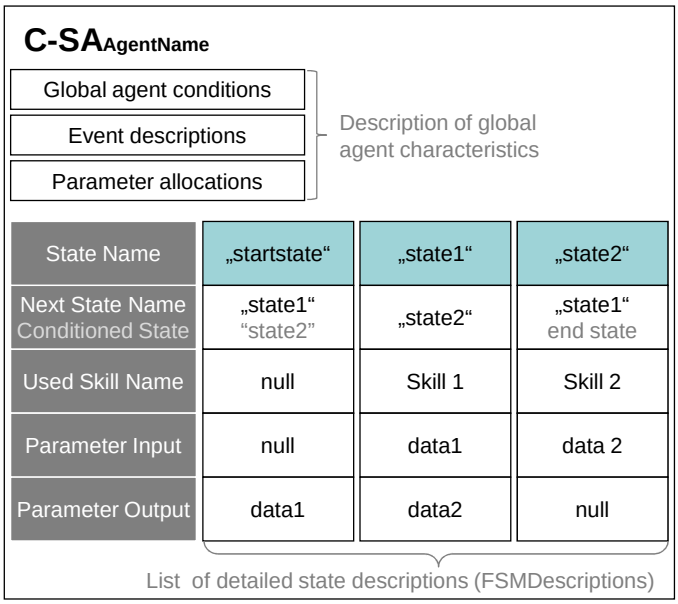


Figure 4.19: Overview of the New Skill Input Data structure.

Following the communication sequence in figure 4.18, the Validator Agent first initialises a Generic Skill Agent in the Reconfiguration Holon. This transformation from the NSID structure to a behaviour-based agent requires the pre-coded skeleton of the Generic Skill Agent to create a specific composite agent. As given in figure 4.20, the Composite Skill Agent provides the standard Skill Agent behaviour and allows to integrate the single components of the New Skill Input Data into a new *AgentAction*. Figure 4.20 represents both transformations from the NSID to the Composite Skill Agent $C-SA_{RH}$ in Reconfi-

guration Holon and respectively the $C-SA_{SH}$ in Standard Holon. The Composite Skill Agent $C-SA$ contains all required Cloned Skill Agents $Cl-SA$ that allow to access the behaviours for the single parts of the new agent. Details and examples of the Finite State Machine and the transformation into the the composite agent are provided in the implementation chapter 6.

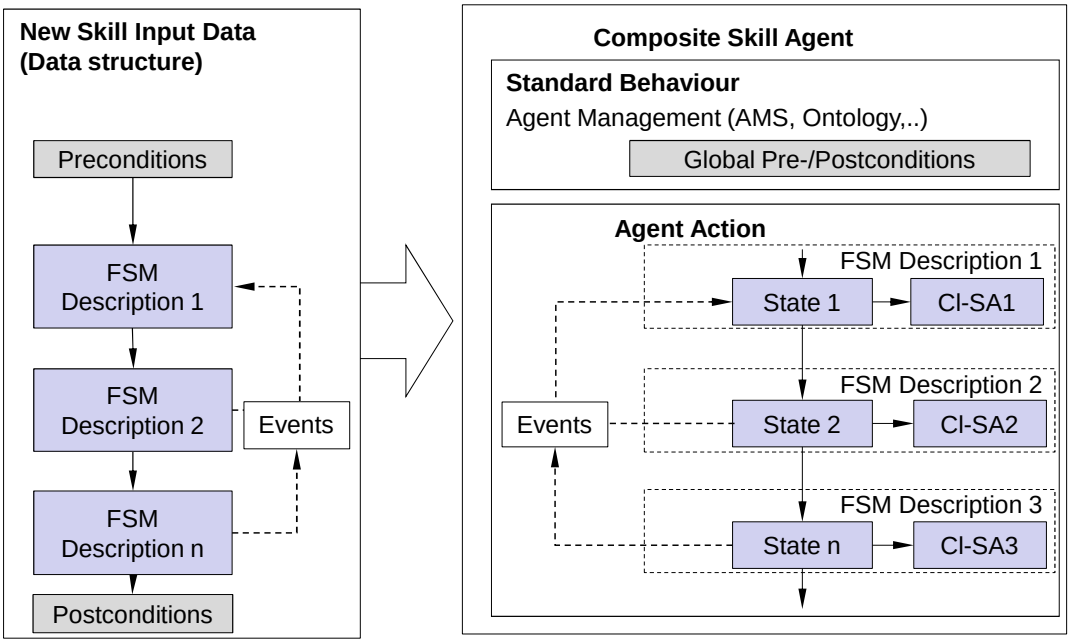


Figure 4.20: Transformation from a New Skill Input Data to a Composite Skill Agent.

Inbetween the generation of the $C-SA_{RH}$ and the $C-SA_{SH}$, the Validity Check executed in real-world has to be prepared as outlined in chapter 5. Only the subsequent execution of this check allows a dependable use of the new service and initiates its activation in Standard Holon.

Section 4.2 to section 4.3 described the composition mechanism within a MobComm reconfiguration. The next section, however, selects the implemented self-organisation as an elementary characteristic of the reconfiguration mechanism, and discusses its strength and weaknesses.

4.4 Integration of Self-Organising Properties

The reconfiguration mechanism described in the last sections contains self-organising properties as required in Research Task 1. As introduced in the literature review in section 2.2.2, self-organisation can be applied by different means and mechanisms in a

multi-agent system. This chapter, however, demonstrates the strength and weaknesses of self-organisation in the presented mechanism.

For the following analysis, self-organisation is specified as introduced in definition 2.12 on page 45, and partitioned into three assessable aspects:

1. Self-management: The system adapts to its environment *without being controlled from the outside.*
2. Structure adaptation: The system *establishes and maintains a certain kind of structure.*
3. Decentralised control: There is *no central point of failure* in the system. Adapted from: [Zadeh, 1963].

The named aspects will be analysed in the following regarding their application in the MobComm reconfiguration mechanism.

Self-Management

The self-management aspect that contains the adaptation to an environment without outside control is based on the MobComm architecture as presented in figure 3.2 on page 77.

The environment of the Reconfiguration Holon is only Standard Holon following the description of the architecture in section 3.4. The integration of Standard Holon-knowledge, the adaptation to ontology changes, and the self-awareness of reconfiguration capabilities are contained in the self-management aspect.

The integration of Standard Holon-knowledge as described in section 4.1 is executed self-managed and independent of the environment. The reconfiguration is both independent of content and complexity of Skill Layer as well as amount of Skill Agents provided in Standard Holon. A dynamic reaction to changes in Standard Holon can be provided by the system. The reconfiguration mechanism is able to flexibly self-adapt to a changing scope and content of already used Skill Agents as long as the agent behaviour is mapped into Preconditions and Postconditions.

For the enhancement of environment adaptation, Standard Holon-knowledge is integrated into the reconfiguration mechanism on demand. Thus not only the quantity of messages and the reconfiguration time can be controlled, but the influence of the Skill Layer complexity on the mechanism's scalability can also be reduced. Even if the according evaluation results are further detailed in the evaluation chapter 7, the impact of Skill Layer complexity on scalability is introduced in the following.

In case a high composition level (e.g. Level 3 of figure 4.13) has to be applied during reconfiguration and a high number of Skill Agents is provided in Standard Holon, the scalability of reconfiguration declines due to the complete mapping of the Cloned Skill Agents to Reconfiguration Holon. Research Task 8 desires a fast adaptability to new processes once changes have been made. However, this can no longer be ensured in the described constellation of composition level and number of Cloned Skill Agents. The detailed evaluation of scalability in Standard and Reconfiguration Holon is given in section 7.4.

This limited scalability results in a decreased self-adaptability of the system to its environment during reconfiguration and thus weakens the applied self-organising properties.

Besides the integration of Standard Holon-knowledge, the adaptation of the reconfiguration mechanism to ontology changes concerns the self-management. The Reconfiguration Holon adapts dynamically to the ontology provided in Standard Holon at the beginning of every reconfiguration process concurrently with the creation of the Reconfiguration Holon. While the initial adaptation to the ontology is fully self-managed, the dynamic integration of new ontology concepts is not supported in the presented reconfiguration mechanism. Its investigation is an extension of the presented work and thus directed to future work in chapter 8.

The last aspect of self-management discussed in this analysis is the system's self-awareness to the applied reconfiguration principles. If self-awareness is fully applied in the system, all occurring reconfiguration situations must be handled as specified by the mechanism. Table 4.4 overviews a set of important situations at the beginning (initial) and at the end (finalising) of a reconfiguration including the actual and the desired reaction of MobComm.

	Reconfiguration situation	Actual reaction	Desired reaction
Initial	No Request Transformation but integration of Events.	Termination of reconfiguration.	Composition mechanism is executed and new skill agent with new Events is created (Future work).
	No Request Transformation and no integration of Events.	No execution of a composition mechanism and termination of the Reconfiguration Holon. The New Skill Description is transferred to Task Layer scheduling.	
Final	Inconsistent/missing parameter allocations.	Termination of reconfiguration.	Application of optimisation strategies including use of former knowledge (Future work).
	Inconsistent sequence of Skill Agents.	No awareness of inconsistencies.	Integration of Validity Check (Chapter 5).

Table 4.4: Reaction to different reconfiguration situations.

Two initial situations are regarded which both deal with the identified *Request Transformation* and the integration of *Events* during reconfiguration. In the proposed mechanism only *Matching Reports* that are based on *Request Transformation* messages are evaluated. Accordingly the integration of new conditional *Events*, as specified in the MobComm Skill definition 3.6, can only be realised if condition inconsistency with *RequestTransformations* occur as well. For consistency to MobComm specifications, the execution of a reconfiguration mechanism is desired for the integration of new conditional scheduling into Skill Layer. If neither a condition inconsistency nor *Events* have to be integrated, no reconfiguration is initiated which also matches with the desired reaction of the system. The used New Skill Description is redirected to the temporal scheduling in Task Layer.

The final reconfiguration situations, however, are related to a dependable use of the new skill as desired in Research Task 3. Even if inconsistent parameter allocations are checked in the proposed mechanism, optimisation strategies are missing and have to be elaborated in future work. The actual termination of reconfiguration provides the avoidance of inconsistencies but further work has to allocate mechanisms that enhance its robustness. In literature, the Layered Learning [Stone, 2007] approach is identified as a possible learning strategy to be integrated in MobComm reconfiguration. Besides the integration of Machine Learning, the implementation of a formal verification framework as proposed in the motion planning algorithms of [Fainekos et al., 2009] (cf. section 2.1.3) has to be investigated. The provision of a mathematical framework is able to avoid inconsistencies by a formal design which is highly recommended for future work.

The presented self-management in MobComm cannot prevent inconsistencies in the resulting Skill Agent during reconfiguration. The verification of this sequence is required for a dependable integration of new Skill Agents (Research Task 3) and motivates the Validity Check as detailed in chapter 5.

Structure Adaptation

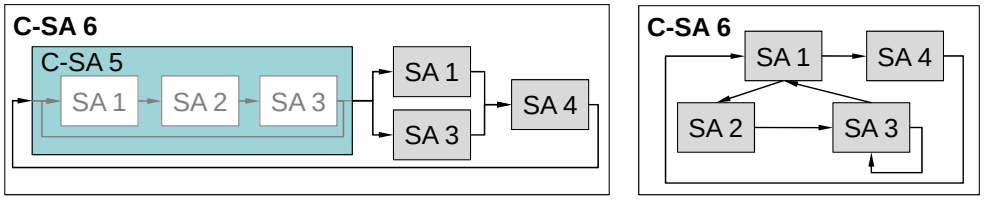
The second feature in the self-organisation analysis is the adaptation of a structure. Within the Reconfiguration Holon, beliefs, desires, and intentions of the used agents are directed to provide a new functionality in the form of a Generic Skill Agent.

Based on the BDI characteristics of reconfiguration agents, the application of a distributed search mechanism allows the generation of the Generic Skill Agent fully self-

managed. The proposed reconfiguration steps use backward search mechanisms that require the compliance of Preconditions during the *RequestCondition* executions. To take the assumption of semantic correctness (cf. section 1.3.3) and a small set of Skill Agents (i.e. between 2 and 30) as given, the presented mechanism provides a solid and robust reconfiguration solution. Rule-based search systems or constraint satisfaction as example alternatives must be regarded in future work to enhance the optimality of the reconfiguration results.

The structure adaptation into a Generic Skill Agent also requires a rule in regards to the reuse of Composite Skill Agents (C-SA) in a subsequent reconfiguration. Due to the basic principle of MobComm, Composite Skill Agents do not differ from the already existing agents. According to that principle, Composite Skill Agents that have been generated in former reconfigurations would be required to be reused in subsequent reconfigurations as well. Further advantages are a flat reconfiguration hierarchy as presented in figure 4.4 and an effective reuse of former reconfiguration knowledge.

In contrast to the compliance of the MobComm principle with a reuse of Composite Skill Agents, the weak controllability of the reconfiguration mechanism is a disadvantage of the reuse. As presented in figure 4.4(a), the content of the Composite Skill Agents is not visible to the environment of the agents. In the example presented in 4.4(a), the skill C-SA5 is only visible as a single service in the multi-agent system independent if linear skills or a complex path structure are contained.



(a) C-SA with reuse of other C-SAs.

(b) C-SA without reuse of other C-SAs.

Figure 4.21: Comparison of a composite agent with and without the reuse of other C-SAs.

This encapsulation that is motivated by the simple and dynamic service allocation of Task Layer reduces the controllability of skill complexity. As discussed in section 3.3, the assumption to allow only linear Task sequences is set to state the distinct focus on a skill-based design. In line with the future enhancement of Task Layer by conditioned state machines, the reuse of Composite Skill Agents has to be facilitated as well. In the

actual MobComm design, complexity and size of Composite Skill Agents are not known during reconfiguration which decrease the system's controllability.

Due to the Research Tasks 3/5 that require a fast and dependable skill integration, the loss of control regarding skill complexity is not acceptable. This leads to an exclusion of the reuse of former composite agents within this work.

Decentralised control

The third self-organisation property is the avoidance of a single point of failure. As the Reconfiguration Holon is mainly structured as a heterarchy, no single point of failure can be identified during the composition mechanism executed by the set of Execution Agents. In contrast to that, the Initiator Agent constitutes a single point of failure during the evaluation of the *Matching Reports* and the transformation into the New Skill Input Data structure. Even if the reconfiguration management violates the self-organisation principle and states a single point of failure, its advantage is the ability to check the consistency of reconfiguration results before their integration in Standard Holon.

The discussion of decentralised control finalised the section about self-organising properties in MobComm. Even if a set of open issues is directed to future work such as the integration of new ontology vocabularies or the enhancement of reconfiguration self-awareness, the proposed mechanism can be classified as self-organising as desired in Research Task 1.

In the following conclusion the reconfiguration mechanism is further analysed regarding its task compliances and future work potentials.

4.5 Conclusion

The last chapter described the proposed skill-based reconfiguration mechanism. The mechanism is divided into three sections that include the creation of the Reconfiguration Holon, the execution of the Distributed Skill Composition, and the Generic Skill Transformations. The analysis of self-organising properties evaluated already the integrated self-organisation as desired in Task 1. In the following, the remaining research tasks are checked as well regarding their compliance in the proposed mechanism. The corresponding summary is provided in table 4.5.

The disturbance of productivity caused by a reconfiguration has to be avoided for the compliance with the second research task. This task can only be fulfilled by both the char-

Research Task	Compliance
Provide a reconfiguration mechanism that realises self-organisation (Task 1).	Yes
Provide a reconfiguration mechanism that does not affect the level of productivity during reconfiguration (Task 2).	Yes
Provide mechanisms that ensure dependability in the use of new functionalities (Task 3).	No
Provide a reconfiguration mechanism that allows hardware abstraction (Task 4).	Yes
Provide a reconfiguration mechanism that is robot configuration independent (Task 5).	Yes
Provide a reconfiguration mechanism that is aware of the limitations of its reconfiguration capabilities (Task 6).	Partial
Provide a reconfiguration mechanism that is open for a broad range of functional process changes (Task 7).	Yes
Provide a satisfactory fast adaptability to new processes (Task 8).	Partial

Table 4.5: Compliance with the set of research tasks by the reconfiguration mechanism.

acteristics of the MobComm architecture as described in chapter 3 and the reconfiguration mechanism. The strict separation of standard processes and reconfiguration mechanism through the holonic paradigm provides the basis while the integration of Cloned Skill Agents into the cognitive multi-agent system allows the final encapsulation of the mechanism in the Reconfiguration Holon. The Generic Skill Transformations, however, also contribute to the creation of a behaviour-based Skill Agent in Reconfiguration Holon which allows it to seamlessly integrate the new Composite Skill Agent into the running process of Standard Holon. The set of characteristics of both the architecture and the mechanism fulfil the requirements of Research Task 2.

The third research task aims for the dependable integration of the new Skill Agent into the running system. As the proposed mechanism applies self-organisation with an analysed lack of self-awareness, as presented in table 4.4, dependability occurs only at a very low level. The verification of parameter allocations is not sufficient for this task compliance that requires an enhanced dependability level realised with the Validity Check detailed in the following chapter.

Research Tasks 4/5 have to also rely on the characteristics of MobComm architecture with the needed independence of robot configuration and hardware abstraction. The flexibility and dynamics of the Standard Holon-knowledge integration is the main contribution of the reconfiguration mechanism to this task. According to figure 4.7 on page 101, the distributed mapping of Skill Agent behaviour in the Execution Agents is the basis for a

dynamic self-adaptation to changing skill specifications and used robot hardware.

Due to the analysis in table 4.4, the proposed reconfiguration mechanism has a lack of self-awareness and provides only basic protection mechanisms against incorrect parameter allocations as introduced in figure 4.16. As a basic but no sufficient handling of reconfiguration self-awareness is given, Research Task 6 can only be accomplished partially.

In contrast to the lack of self-awareness, a high degree of functional reconfigurability can be provided as desired in Research Task 7. The semantic content of the New Skill Description and the composition of existing skills is not limited in content or complexity within the proposed mechanism which allows the integration of a broad range of functional changes into a running MobComm application.

The compliance with a short reconfiguration time as handled in Research Task 8 is dependent on the weak scalability of the mechanism. As the scalability of the system deceases with the number of used Skill Agents, the proposed mechanism has to be applied to a small amount of Skill Agents to reach an acceptable reconfiguration time. Due to this constraint, only a partial fulfilment can be stated for Research Task 8.

The missing compliance of a dependable skill integration that is mapped into Research Task 3, is required to be further elaborated on in this work and motivates the Validity Check in the next chapter.

Chapter 5

Validity Check

To enhance dependability in MobComm as analysed in section 4.5, a Validity Check is proposed as a termination of MobComm reconfiguration in this chapter. As the Validity Check is introduced to increase dependability according to Research Task 3, the according term is defined in the following:

Definition 5.1 (Dependability) *A multi-agent system is dependable if it maintains its service according to specifications even if disturbances occur that are due to events endogenous to the system such that reliance can justifiably be placed on this service [Lockemann and Nimis, 2009].*

In the proposed Validity Check, the improvement of dependability targets especially the avoidance of unwanted and harmful behaviour that are defined as follows:

Definition 5.2 (Unwanted Behaviour) *Unwanted behaviour occurs locally in the system if the specification made in the New Skill Description differs from the outcome of the reconfigured Composite Skill Agent, and if the Composite Skill Agent endangers the productivity of the total robot system.*

Definition 5.3 (Harmful Behaviour) *A behaviour is harmful if - regardless the actions of the human- the safety of the worker is compromised in any way.*

Based on these definitions, a behaviour analysis is conducted to identify unwanted or harmful aspects of the behaviour of the reconfigured Composite Skill Agent in Standard

Holon. Additional to the functional matching of the resulting Skill Agent with the inserted New Skill Description, the focus of the following analysis is on a proper skill transformation and the correct agent behaviour of the new skill in Standard Holon.

5.1 Behaviour Analysis

The subsequent behaviour analysis is performed to identify misbehaviour in the Composite Skill Agent in the categories safety, software, and functionality. The aimed outcome of this analysis is a set of targeted actions that enhance dependability. The resulting actions further specify the Validity Check as proposed in section 5.2.

Starting with the standard Skill Agent behaviour, figure 5.1 gives a schematic view of the skill's activation and termination.

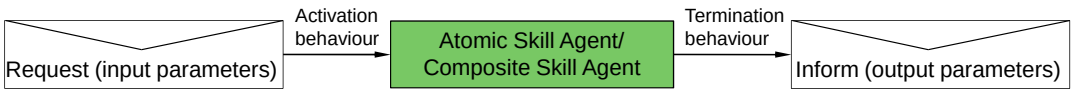


Figure 5.1: Standard activation and termination behaviour of any Skill Agent.

For the initialisation of any Skill Agent, an activation message in the form of a *Request*-message is required. Equally to that, an *Inform*-message states the termination of a skill execution. As introduced in section 4.3, the Skill Agent itself is viewed as a black box from the outside and accessible via a registered service.

Thus, the following Skill Agent activation and termination rules can be stated:

- Skill Agents can only be activated correctly if they receive a FIPA *Request*-message [FIPA, 2005]. They have been determined properly if they reply with a FIPA Inform Message.
- Skill Agents can only run as specified if the set of incoming and outgoing parameters is complete and in the correct order.

Based on this standard behaviour, the structure of the Composite Skill Agent is further analysed to detect its specific behaviour resulting from the preceding Generic Skill Transformation.

Following the explanation of the Generic Skill Transformation in section 4.3, a Composite Skill Agent contains a Finite State Machine that is based on the skeleton of a Generic Skill Agent. The New Skill Input Data structure specifies a sequence of Cloned Skill

Agents that is integrated in the Generic Skill Agent and results from the reconfiguration process.

Coupled with the cloned skill integration, the required resource allocation is also included in the Composite Skill Agent. As the connection between Skill Layer and Resource Layer is based on a static service allocation (cf. figure 3.5 on page 80), the resource allocation inside the Composite Skill Agent is completely inherited from the cloned Skill Agents. Consequently, Resource Layer activation and resource allocation constitute the cloned agent behaviour in the Composite Skill Agent as presented in figure 5.2. The Atomic Skill and Resource Agents that are the origin are pre-coded and integrated following a software testing process. After a short introduction of the levels presented in figure 5.2 in the following, these levels are further detailed below.

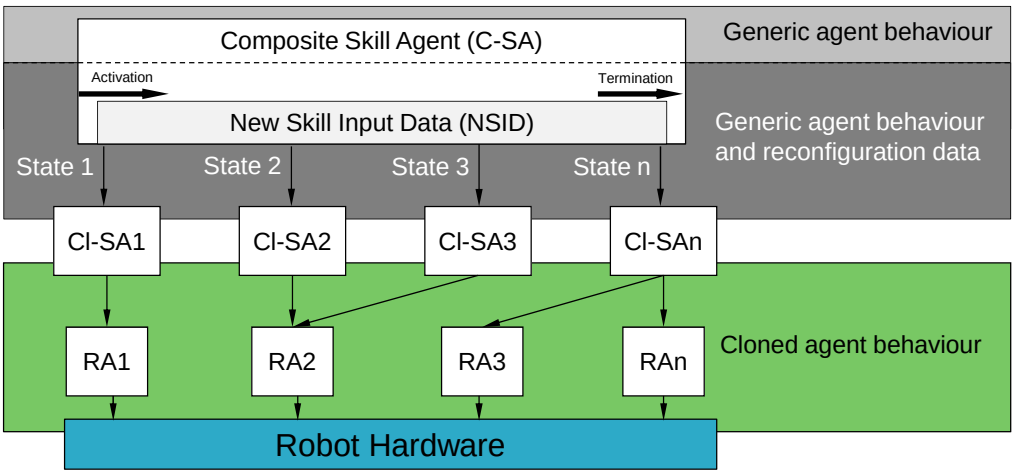


Figure 5.2: Classification of the Composite Skill Agent structure in cloned respectively generic behaviour and reconfiguration data.

As presented in figure 5.2, the generic agent behaviour of the Generic Skill Agent skeleton is enriched with specific reconfiguration data in the form of the New Skill Input Data.

The behaviour classification performed in figure 5.2 constitutes the basis of the analysis of unwanted or harmful behaviour in the following. The generic and cloned agent behaviours display three classes of unwanted respectively harmful behaviour as pictured in figure 5.3.

As the safety level is related to the real-world environment of the system, harmful behaviour may occur where the safety for objects (no collision) or humans (no imperilment) in the robot environment cannot be guaranteed. Within the Composite Skill Agent,

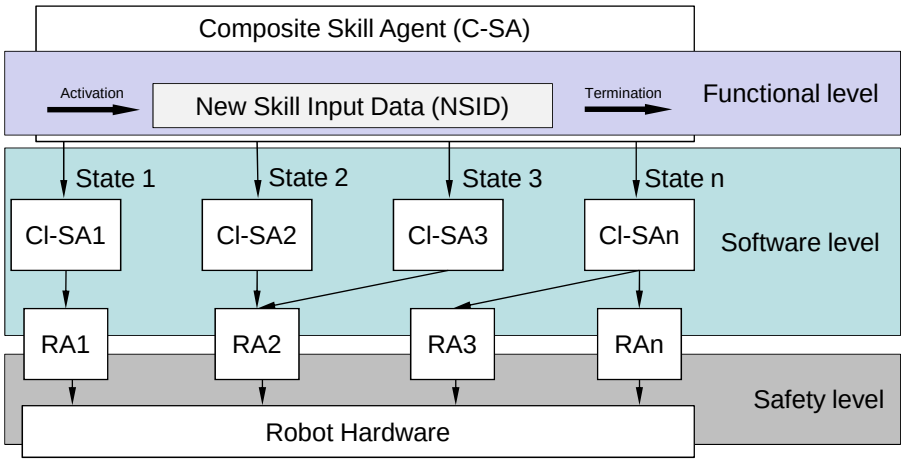


Figure 5.3: Levels in the behaviour analysis of the Composite Skill Agent.

the safety level comprises the robot hardware and the interface to Resource Layer which is composed of cloned agent behaviour. Even if the agent behaviour itself is inherited, its combination and environmental connection can result in the undesirable harmful behaviour.

The software level, however, may produce unwanted behaviour and is related to classical software correctness (e.g. deadlock-free system) and agent-specific software parameters (e.g. activation and termination messages). The software-specific verification of the Composite Skill Agent must check both content and structure of the created Finite State Machine within the agent.

The functional level is additionally allocated to the risk of unwanted behaviour as it focuses on the conformance of the inserted New Skill Description with the resulting functionality in the Composite Skill Agent. Content, order, and Event integration in the New Skill Input Data have thus to be checked to avoid any unwanted behaviour at this level.

After a short introduction of the three levels, their analysis regarding the reduction of misbehaviour in a Validity Check is conducted in section 5.2.

Safety Level

In general the required safety level of a robot system, especially regarding static industrial manipulators, is strictly regulated in a set of national and European norms. The most important norm for mobile and collaborative robot systems, however, is EN ISO 13849 that replaces the formerly important norm EN 951/1 [International Organization for Standard-

ization, 2006]. This document handles the safe collaboration between a robot system and a human worker. A safe robot has to show a minimum of EN 951/1 category 3 control. An example of safety policies that regulate a static but collaborating robot system are given in [Schraft et al., 2005]. As overviewed in figure 5.4, a collaborating static manipulator contains a safe part that monitors velocities whereas the gripper and force torque sensor are unsafe and follow only category B.

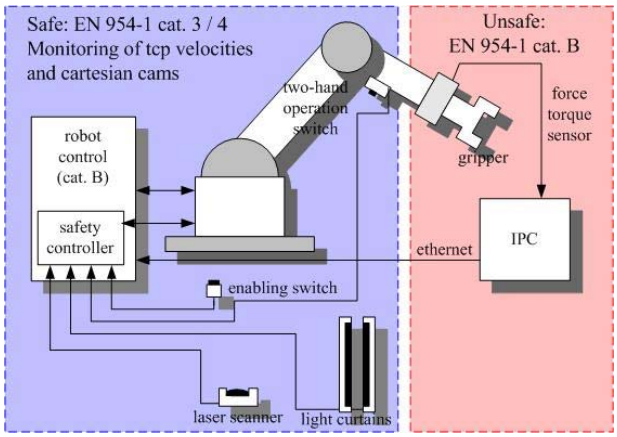


Figure 5.4: Collaborating robot system regulated by safety policies. Source: [Schraft et al., 2005].

The facilitation of a safe robot operation is investigated and discussed widely in literature. [Frei et al., 2007c] states that behaviours that are not pre-programmed [but emerged] such as the proposed Composite Skill Agent must be checked for correspondence to safety policies in different levels of measurement:

1. Mechanical level such as the mechanical blocking of certain movements.

2. Control level such as the limitation of speed of axis movements.

3. Software level such as the prohibition of certain configurations.

The safety policies, however, that are addressed in [Schraft et al., 2005] belong to the mechanical and control levels as specified in [Frei et al., 2007c] with an implementation in the robot control structure. As the safety norms address or at least require the characteristics of the control level, the implementation of a safe MobComm needs a formal description mechanisms as for example hybrid controllers using Linear Temporal Logic. As overviewed in the literature review in section 2.1.3, the Linear Temporal Logic is based on the specification of discrete event models for the robot environment. By the continuous

definition of safety policies that correspond to the control laws as for example presented in figure 2.5 on page 25, a high level planner can reason on these formulae and guarantee a system behaviour following a specification. The development of a formal mathematical model that describes the MobComm requirements is directed to future work.

A safe industrial mobile robot is the requirement to apply this automation technology without ethical concerns regarding man-machine interaction. Even if the reconfiguration mechanism is beyond the influence on the robot actuation, as overviewed in figure 7.7 on page 185, the integration of an adaptable robot behaviour that reacts to the characteristics of a human companion is desired in future work. As described in [Angerer et al., 2012], an adaptive robot behaviour that is dependent on the individual movements of a worker can be integrated by observing the human and generating automated motion sequences out of the detected behaviour (e.g. following [Kulic and Nakamura, 2010]). The adaptive robot behaviour can further enhance the acceptance of industrial mobile robots in car factories and thus dispel any ethical concerns regarding the use of safe robots in industry.

Software level

In contrast to the safety level that concentrates on the avoidance of harmful behaviour, the software level is focused on the prevention of unwanted behaviour by proving classical and agent-based software parameters. Software functionality that is robot control specific cannot be covered by this level as MobComm has no access to these control structures. This limitation results from the assumption set in section 1.3.3 and can only be mitigated by the provision of a mathematical robot description and its on-line verification.

Following figure 5.2 and figure 5.3, the software level includes both generic and cloned behaviour additional to the use of reconfiguration data.

As all used agents, including the Generic Skill Agent skeleton, are pre-programmed or cloned, the verification in this level concentrates primarily on the correct activation and termination of the agents. The order and content of activation and termination messages finally constitute the resulting functionality of the Composite Skill Agent.

Thus, the global activation and termination messages of the Composite Skill Agent and the agent-internal activation and termination of the Cloned Skill Agents are the main sources of misbehaviour at this level as presented in figure 5.3. By proving the right order and content of the activation and termination messages unwanted behaviour like an unsequenced robot movement can be avoided at software level.

Functional Level

In common with the software level, the functional level contains as well the risk of unwanted behaviour. The key function of this level is the validation of the New Skill Input Data. Following definition 5.2 of unwanted behaviour, the resulting functionality in the Composite Skill Agent has to comply with the specifications given in the New Skill Description. Consequently, an erroneous sequence of Cloned Skill Agents in the Finite State Machine, the incorrect selection of the Cloned Skill Agents, and a deficient integration of Events constitute the risk of unwanted behaviour. The final effect of the new functionality is the activation of the robot actuation in real-world.

The sequence and choice of cloned skills in the New Skill Input Data cannot be satisfactory verified by a Validity Check that is executed after the reconfiguration process. The effort to validate such a data structure that was built by the application of self-organisation, as described in section 4.4, would consume the gained advantages of self-organisation (Research Task 1) and the maintenance of productivity (Research Task 2).

In ADACOR [Leitão et al., 2006], the interaction between the used holons is performed by synchronising individual Petri net models providing a fully developed formal behaviour verification during reconfiguration. As MobComm does not provide a formal verification of reconfiguration results, the final result which is the impact on the real-world is checked. Therefore, unwanted behaviour cannot be avoided but merely detected by a traced execution of the new Composite Skill Agent in real-world.

For an immediate functional validation of reconfiguration results, a reconfiguration control is investigated within the scope of future work. Even if the Initiator Agent constitutes a single point of failure in MobComm as exposed in section 4.4, the mechanism does not provide a central reconfiguration control. Literature gives different approaches for a control of reconfiguration activities. A controller agent is for example proposed in [Mani et al., 2008] with the awareness of reconfiguration knowledge and functionalities. A further approach for the functional validation of agent behaviour is given in [Zhu, 2001] where the definition of scenario descriptions permits a central behaviour verification.

To apply a formal verification to MobComm, the core requirements of the proposed system have to be mapped into logic formulae. An initial set of requirements covers a three dimensional environment description, a formal model of the mobile robot including the kinematic of the manipulator, a list of safety invariants (e.g. speed of axis movements), and finally state-chart diagrams of the required Process, Task, and Skill Agents. These

requirements are regarded as an initial input set for a formal verification mechanism in future work.

Through the application of a behaviour verification, as proposed in literature, the functional level and thus the system's dependability can be further enhanced. As the mutual influence between a verification tool and the compliance of established research tasks is investigated insufficiently, their coherence must be further examined to enhance dependability without drawbacks in the set research goals.

Even if the functional level cannot be covered completely within this work, a traced execution of the new Composite Skill Agent allows us to detect functional misbehaviour before the permanent integration of the new functionality in Standard Holon.

Implication of the Validity Check

Once the Composite Skill Agent has been analysed regarding potential risk of unwanted or harmful behaviour, a Validity Check that is executed after the reconfiguration mechanism is inferred in the following with the goal to enhance dependability of the reconfiguration result.

As described above, safety level contains important activities for the dependability at hardware control level. The investigation of a safe MobComm system that complies with European standards has to include safety rules and system invariants as proposed in [Frei et al., 2007c]. The integration of safety rules for MobComm is integrated in future work as given in table 5.1.

At software level, however, the activation and termination behaviour of the Composite Skill Agent and its corresponding Cloned Skill Agents are verified. For this verification, the bases of comparison are the global Pre- and Postconditions for the global agent behaviour and the parameter allocations built during the Distributed Skill Composition (cf. section 4.2) for the internal behaviour. For the execution of this verification, the global and local activation respective termination messages are captured by a real-world sniffing mechanism. Sniffing in this case means that the network traffic in Standard Holon is logged for all agent interaction.

Even though the investigation of a reconfiguration controller is directed to future work at the functional level, the real-world impact of the new functionality can be validated by means of this Validity Check. Through the provision of a resource schema for the Composite Skill Agent by the Reconfiguration Holon, the real-world impact is checked by

<table>
<tr><td></td><td>Operation</td><td>Required reconfiguration data</td><td>Realisation</td></tr>
<tr><td>Safety level</td><td colspan="2">Safety Rules, System invariants</td><td>Future work</td></tr>
<tr><td rowspan="2">Software level</td><td>Internal behaviour</td><td>Parameter allocations</td><td rowspan="3">Real-world validation by sniffing</td></tr>
<tr><td>Global behaviour</td><td>Global Pre-/Postconditions</td></tr>
<tr><td rowspan="2">Functional level</td><td>Impact on real-world</td><td>Resource schema</td></tr>
<tr><td colspan="2">Reconfiguration controller</td><td>Future work</td></tr>
</table>

Table 5.1: Proposed activities for a MobComm Validity Check including the software and functional level that are implemented in this work.

sniffing the agent communication in Standard Holon during its execution.

Based on the proposition in table 5.1, software and functional level can be covered in this thesis through implementation of a real-world sniffing mechanism that includes the verification of internal and global agent behaviour additional to the impact of the Composite Skill Agent on its environment.

5.2 Validity Check Design

As a first step in the proposed Validity Check, the Composite Skill Agent is migrated from the Reconfiguration Holon ($C-SA_{RH}$) to Standard Holon ($C-SA_{SH}$) as explained in section 4.3. Even though integrated in Standard Holon, the Composite Skill Agent is not yet permanently usable at Standard Holon-DF, as it is still restricted in use during the Validity Check.

The Validity Check itself is executed by a specific Process Agent for Validity Check, called VC-PA. This Process Agent is initiated by the Validator Agent in Reconfiguration Holon and registered in Standard Holon while the interface to the Reconfiguration Holon is preserved by the Validator Agent. The Validator Agent is the source of reconfiguration data such as the parameter allocations or the resource schema as overviewed in table 5.1 and described in figure 5.5. Based on this reconfiguration knowledge, the sniffing algorithm can be executed by the VC-PA to validate the behaviour of the Composite Skill Agent.

As given in figure 5.5, the VC-PA registers itself at the Standard Holon-DF before asking the running Process Agent $PA_{running}$ for permission to execute the Validity Check. As the VC-PA includes a corresponding set of scheduling parameters, $PA_{running}$ is able to match the temporal conditions with the actual scheduling forecast. In case of a positive

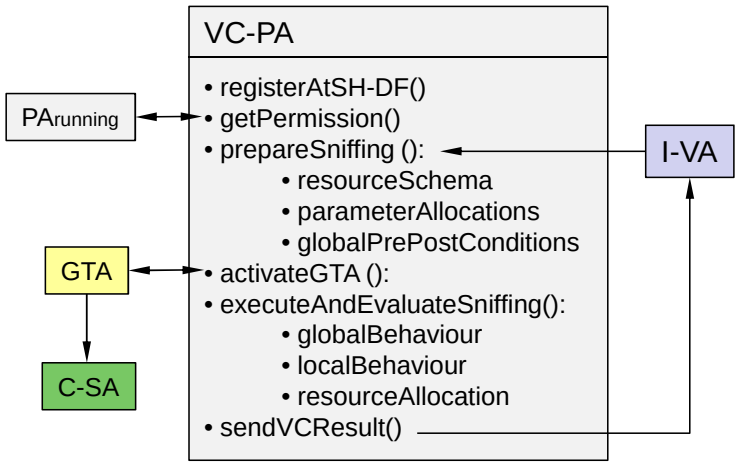


Figure 5.5: Activities of the VC-PA during Validity Check execution.

outcome, $PA_{running}$ has to remain idle and is not able to initiate any Task Agent until the termination of the Validity Check. Standard process execution in Standard Holon can thus not be maintained during the Validity Check.

Once the Composite Skill Agent is registered at Standard Holon-DF, the execution of the new service and its evaluation by means of a sniffing mechanism is initialised by the VC-PA. The temporary integration of the Composite Skill Agent, as presented in figure 5.6, allows the validation of the local and global agent behaviour additional to the tracing of the real-world impact.

During the sniffing mechanism, the total agent interaction of Standard Holon is captured. The principle of message sniffing is established in diagnosis tools such as the Rockwell Automation Java Sniffer [Rockwell Automation Inc., 2006] or the ACLAnalyser as presented in [Botía et al., 2004].

The structure of Standard Holon including the Reconfiguration Holon-interface during message sniffing is overviewed in figure 5.6 and presents the activation of the Validity Check in addition to the set of captured messages during the sniffing mechanism. According to the stoppage of standard process execution, only the agents required for the new Composite Agent are expected to communicate in Task, Skill, and Resource Layer.

The set of captured messages is thereafter analysed to detect unwanted behaviour at software and functional level as analysed in section 5.1. The sniffing is possible as the agent interaction in Standard Holon is strictly protocol-specific as explained in figure 3.5 on page 80. Thus, for the execution of the sniffing algorithm, an internal sniffing schema is created in VC-PA as a basis for the successive evaluation. The resulting schema, as

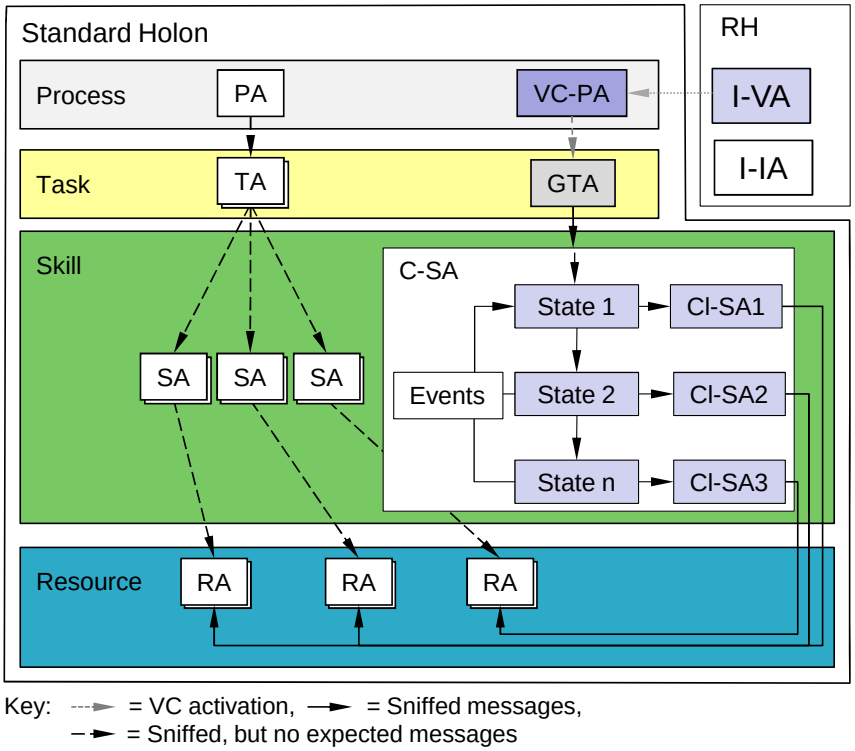


Figure 5.6: Interaction structure of Standard Holon during Validity Check.

presented in figure 5.7, is comprised of a global behaviour part, a local behaviour part, and a real-world impact part.

The global behaviour part is based on the provided Pre- and Postconditions and evaluates the sniffed activation and termination messages that are sent respectively and received by the Generic Task Agent as pictured in figure 5.7. Only if the parameters contained in these messages match with the supplied global Pre- and Postconditions, can the global behaviour part in the sniffing mechanism be validated.

In the local behaviour part, however, the sniffing algorithm covers the internal messaging within the Composite Skill Agent that is not visible to Standard Holon following the final integration of the Composite Skill Agent. By using the supplied parameter allocations, as described in section 4.2.3, the parameters of the sniffed messages between the Cloned Skill Agents are compared to the given reconfiguration knowledge. Only if the comparison results in a match can the local behaviour part be approved by the sniffing mechanism.

The main part of the sniffing mechanism constitutes the evaluation of the impact on the real-world environment by using the resource schema generated during reconfiguration.

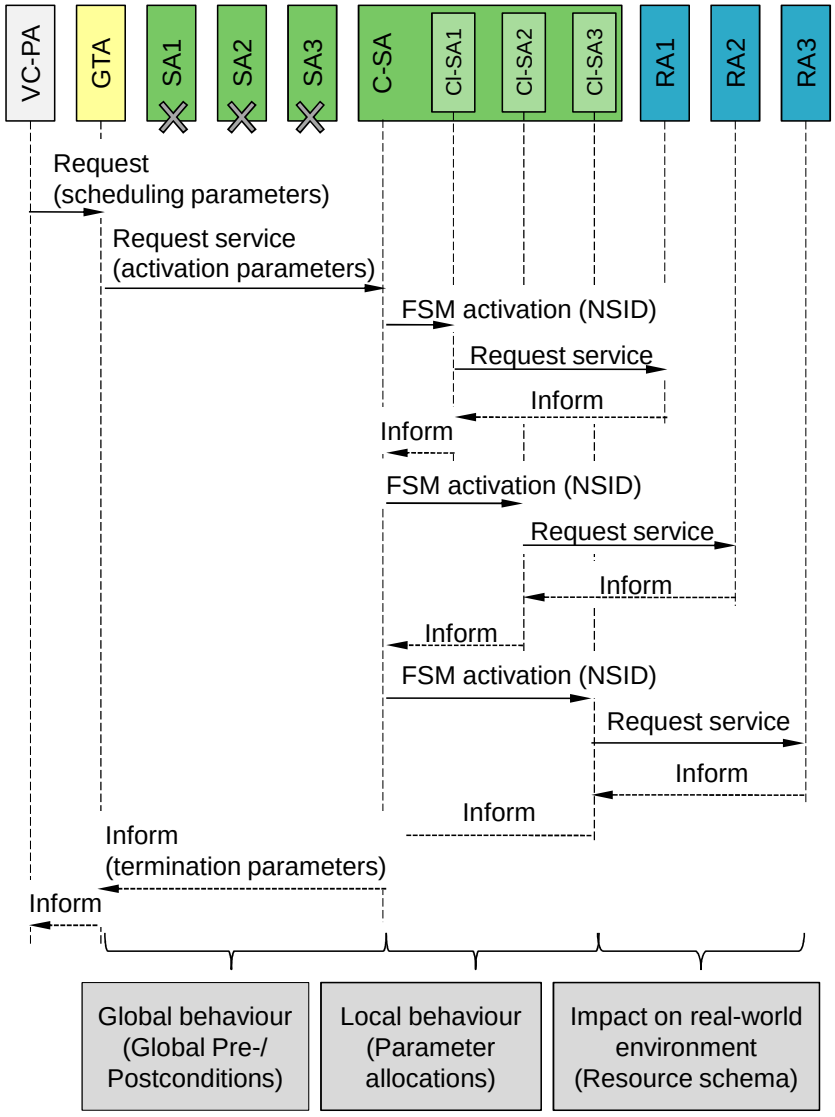


Figure 5.7: Sniffing schema for the validation of the Composite Skill Agent including a local behaviour, a global behaviour, and a real-world impact part.

The specific activation of the Resource Agents results from the sequence of Cloned Skill Agents used in the Composite Skill Agent. For the real-word impact verification only sender, receiver, and message type are evaluated as the message content for the resource allocation is cloned from the Atomic Skill Agents.

Only if the three areas can be approved during message sniffing is the Composite Skill Agent permanently integrated in Standard Holon associated with a successful termination of reconfiguration.

As shape and behaviour of the Composite Skill Agent are identical to the Atomic Skill Agents, the integration of the composite agent contains only the permanent service

registration at the Standard Holon-DF. This registration allows Task Layer to access the service of the new agent through the Agent Management System. The Standard Holon, resulting from the execution of a MobComm reconfiguration, is presented in figure 5.8.

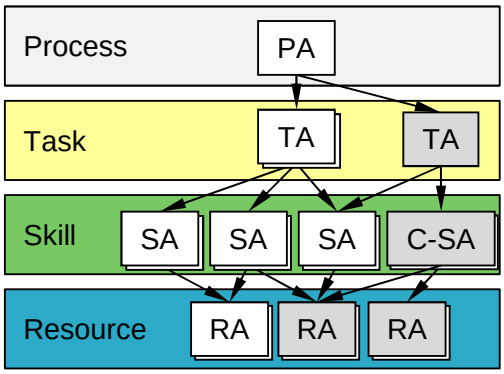


Figure 5.8: Standard Holon after the execution of a MobComm reconfiguration.

The design and execution of the proposed Validity Check is discussed in the following in regards to its compliance with the affected research tasks.

5.3 Conclusion

A suitable validation of MobComm reconfiguration results has to be applied in safety, software, and functional level as overviewed in table 5.1. As the implementation of safety level goes beyond the scope of this thesis, software and functional level are covered in the presented Validity Check. According to that, a sniffing mechanism, that is dependent on reconfiguration knowledge provided by Reconfiguration Holon, is proposed for the validation of the global and local agent behaviour additional to the impact on real-world.

Compared to a solely simulation-based functionality check, as for example applied in SIARAS [Bengel, 2007] and evaluated in the literature review in section 2.2.3, the proposed Validity Check reflects more precisely the real-world conditions before the agent is finally integrated in Standard Holon. Uncertainties based on assumptions in a simulation environment are thus avoided from the outset and allow to enhance the dependability of the new skill integration (Research Task 3). As highlighted in table 5.2, the dependability can be increased from *no compliance* to a *partial compliance* due to the implementation of the key aspects in software and functional level.

In contrast to the enhancement of dependability, the execution of the Validity Check in real-world diminishes the productivity of the standard process for a limited and restricted

Research Task	Compliance
Provide a reconfiguration mechanism that realises self-organisation (Task 1).	Yes
Provide a reconfiguration mechanism that does not affect the level of productivity during reconfiguration (Task 2).	**Partial**
Provide mechanisms that ensure dependability in the use of new functionalities (Task 3).	**Partial**
Provide a reconfiguration mechanism that allows hardware abstraction (Task 4).	Yes
Provide a reconfiguration mechanism that is robot configuration independent (Task 5).	Yes
Provide a reconfiguration mechanism that is aware of the limitations of its reconfiguration capabilities (Task 6).	Partial
Provide a reconfiguration mechanism that is open for a broad range of functional process changes (Task 7).	Yes
Provide a satisfactory fast adaptability to new processes (Task 8).	Partial

Table 5.2: Summary of the compliance of research tasks with an emphasis on tasks modified by the Validity Check.

period of time. Even if this controlled loss of productivity contributes towards a dependable long-term maintenance of productivity in cycle time, a suspension of standard process execution reduces the compliance of Research Task 2 to a partial fulfilment as overviewed in table 5.2. To keep the loss of productivity to a minimum, the proposed Validity Check is executed apart from time-critical standard processes with a preference for production breaks that occur regularly in a shift.

As a summary of the research tasks evaluated in chapter 3 and chapter 4, table 5.2 overviews all compliances with an emphasis on the research tasks modified by the introduction of the Validity Check.

Even if the proposed real-world Validity Check covers only software level and functional level, it constitutes the basis for the dependable usage of MobComm in an industrial environment by balancing the level of productivity with the dependability in handling reconfiguration results. Besides a future integration of safety level in MobComm, the compliance of industrial standards in MobComm is desired to be enhanced in the future with an optimised scheduling algorithm in the VC-PA for a reduction of the productivity losses during the Validity Check.

Chapter 6

Use Case and MobComm Implementation

The following chapter focuses on the implementation of the MobComm reconfiguration as presented in chapter 3 to chapter 5. After the presentation of the use case *Follow transport cart*, the description encompasses the architecture and agent framework in section 6.2 in addition to the reconfiguration mechanism in section 6.3.

6.1 Use Case

To provide a deeper understanding of a typical application for industrial mobile robots in car manufacturing and the specific usage of MobComm, figure 6.1 pictures a process change within a commissioning process executed through the use of an industrial mobile robot system and the presented reconfiguration mechanism.

Figure 6.1(a) overviews the semi-automated commissioning of cardan shafts. Different varieties of cardan shafts are delivered by suppliers at the car plant with the purpose to be sequenced in assembly order of the cars in the production line.

The used mobile robot is able to pick different sorted components and brings them to the required production sequence. The worker still cares for recyclables and empty boxes. As there is a physical distance between the commissioning area and the production line, a driver has to deliver the sequenced parts to their final installation points.

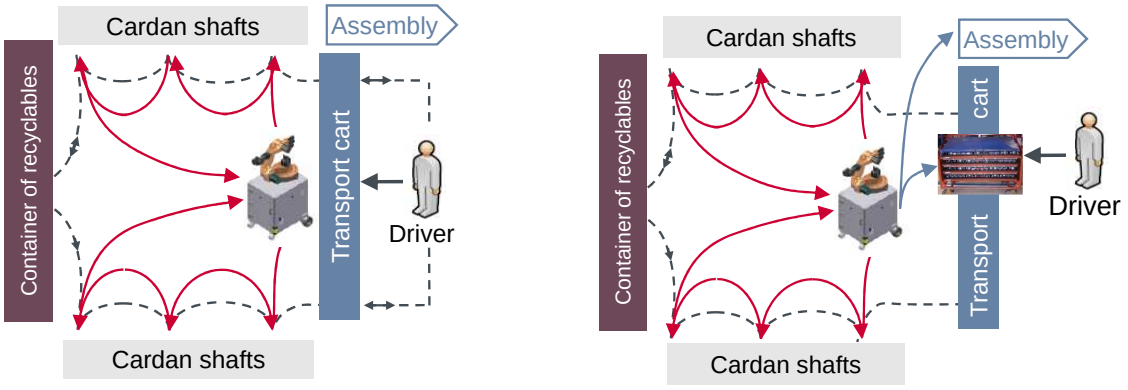
For the standard process *Commission cardan shafts* (cf. figure 6.1(a)), the protocol-specific communication is given in figure 6.2. The responsible Task Agent TA_{comm} initiates

a FIPA Request protocol to pick the cardan shafts from a box. While the Skill Agent SA_{pick} and the Resource Agent RA_{arm} are involved in the picking process, the movement to the transport cart is executed in cooperation of the Skill Agent $SA_{moveLocation}$ and the Resource Agent $RA_{platform}$. For the deposition of the cardan shafts, subsequently the Resource Agent RA_{arm} is allocated again by the Skill Agent $SA_{deposit}$. The temporal scheduling of the commissioning process is controlled by the Task Agent TA_{comm} following the Task Layer specification in the architecture description in section 3.3.

The built robot system, as presented in the experimental setup in section 7.3, provides the flexibility to execute the proposed reconfiguration mechanism for the process change as given in figure 6.1(b). Control-based navigation flexibility, hardware-based gripping flexibility, and a basic set of sensor equipment allows the execution of different system configurations with the provided industrial mobile robot.

For the process change of figure 6.1(b), the new robot functionality *Follow transport cart* is required in the system whereas the reconfiguration is desired to be executed without disturbing the running commissioning process. The process change constitutes an additional transport of the cardan shafts to their installation location at the assembly line. As these locations may vary for different series of models, the transport should be made by following the human-driven transport cart in front.

To trigger the desired process change, an additional Skill Agent $C{-}SA_{follow}$ has to be provided by Standard Holon that allows to track an *EnvObject* and to follow it. For the description of this new functionality the operator inserts a *Reconfiguration Element* in the graphical interface. The mobile robot is required to *move* wherever a specific *EnvObject*



(a) Commissioning of cardan shafts.

(b) Additional transport to the assembly line.

Figure 6.1: Standard process and reconfiguration task in the use case.

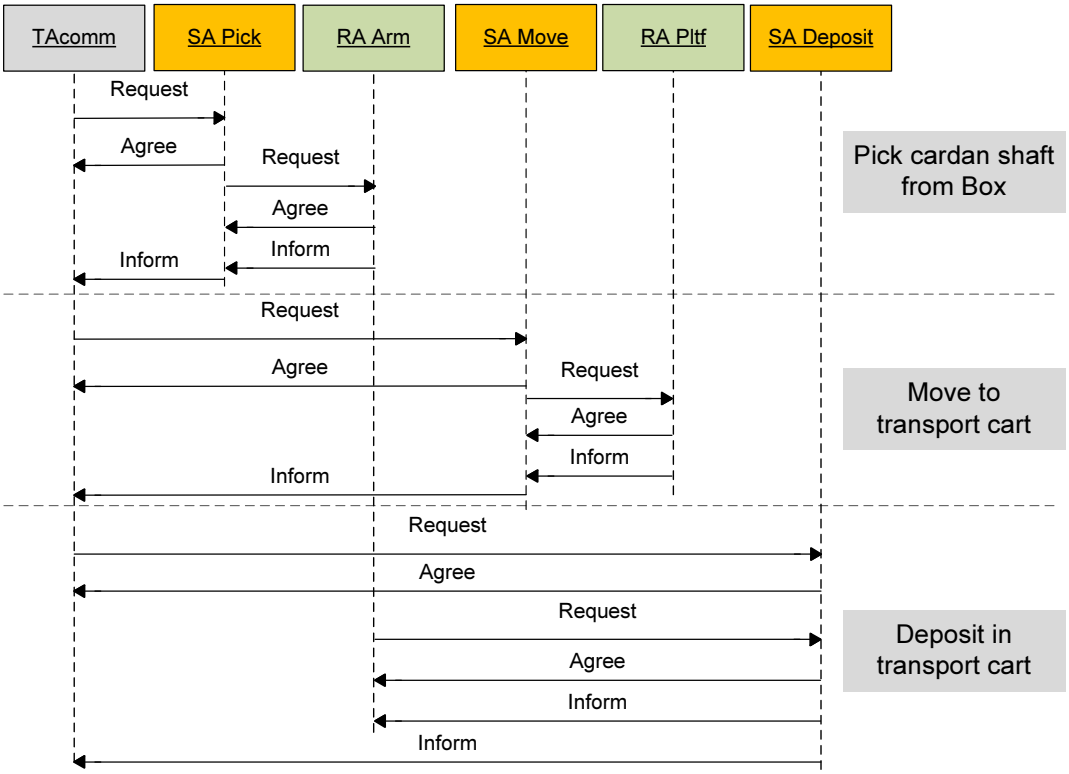


Figure 6.2: Communication in Standard Holon during the commissioning of cardan shafts.

is located until the execution is stopped by the Task Layer. The corresponding Reconfiguration Element is set as follows:

Used skill: *MovePlfm*, Precondition: *MoveLocation*, Value: *EnvObject*, Event: *Loop*.

Once the reconfiguration mechanism is prepared by creating the Reconfiguration Holon, the composition level that corresponds to the New Skill Description is selected. Due to the missing supplier skill, the second level is initiated by the creation of the Execution Agents for the total set of Atomic Skill Agents.

According to the goal/plan-tree of the Initiator Agent, as presented in figure 6.16 on page 155, a *ConditionRequest* with a required *EnvObject* is sent to the Execution Agent $I-EA_{movePlfm}$. As $I-EA_{movePlfm}$ is not able to follow this request with its Precondition *Location* instead of the required *EnvObject*, a *RequestTransformation*-message is sent to the set of Execution Agents with the desire of an *EnvObject* as a Precondition and a *Location* as a Postcondition. The corresponding *RequestTransformation*-message is set as follows:

RequestTransformation(Action: MovePlfm, Provided: *Location*, Desired: *EnvObject*)

By following the list of Pre- and Postconditions of Atomic Skill Agents in table 4.3 on page 106, $I-EA_{detect}$ is able to comply with the sent *RequestTransformation*-message. The communication in the Reconfiguration Holon during reconfiguration is overviewed in figure 6.3. The *RequestTransformation*-message that is accepted by $I–EA_{detect}$ is presented in green in contrast to the refused *RequestTransformation*-messages in red colour.

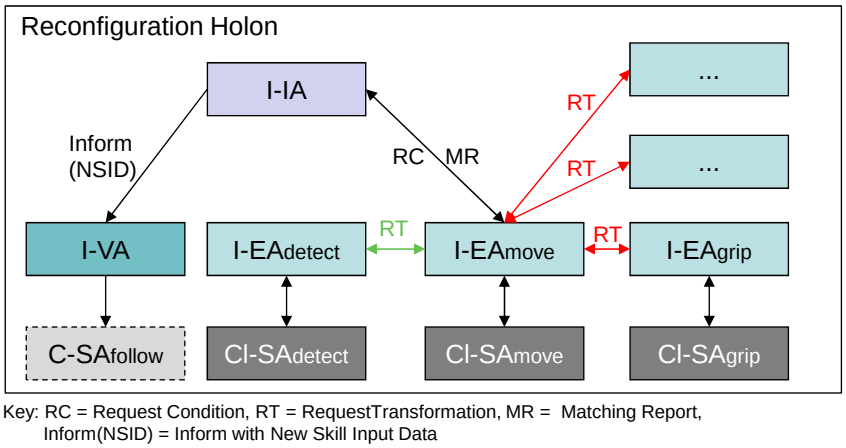


Figure 6.3: Communication of Reconfiguration Holon during use case.

According to the specification of the reconfiguration mechanism, the Execution Agents $I-EA_{movePlfm}$ returns a *Matching Report* to the Initiator Agent to state the termination of the Distributed Skill Transformation. The processed *Matching Report* finally turns into the New Skill Input Data as pictured in figure 6.4. Two *FSMDescriptions* are created that can be repeated due to the integrated *Loop-Event*. Finally, the Validator Agent integrates the new Composite Skill Agent $C-SA_{follow}$ into Standard Holon to prepare the execution of the Validity Check.

To be able to execute the Validity Check of the presented use case, a test environment is built in the laboratory as presented in figure 6.5. The requirements of the standard Task Agent TA_{comm} for the commissioning of cardan shafts are supplied by a box of sorted components and the according transport cart. This transport cart, however, is also used for the execution of the Validity Check for $C - SA_{follow}$.

In order to be able to validate the Composite Skill Agent $C-SA_{follow}$, the following instance of an *EnvObject* has to be provided to specify the required *TransportCart*:

Class: *EnvObject*, Name: *TransportCart*, Height: *1,8 m*, Width: *1,5 m*, Shape: *box*

This information is necessary as the Composite Skill Agent $C–SA_{follow}$ has the generic

C-SA$_{follow}$ New Skill Input Data | Event: Loop

„startstate“	„state1“	„state2“
„state1“	„state2“	„state1“ end state
null	SA$_{detectLocation}$	SA$_{move}$
null	envObject	Location
EnvObject	Location	null

Figure 6.4: New Skill Input Data of the use case *Follow transport cart*.

functionality to follow any *EnvObject* and requires a concrete instance to execute the Skill Agent in real-world.

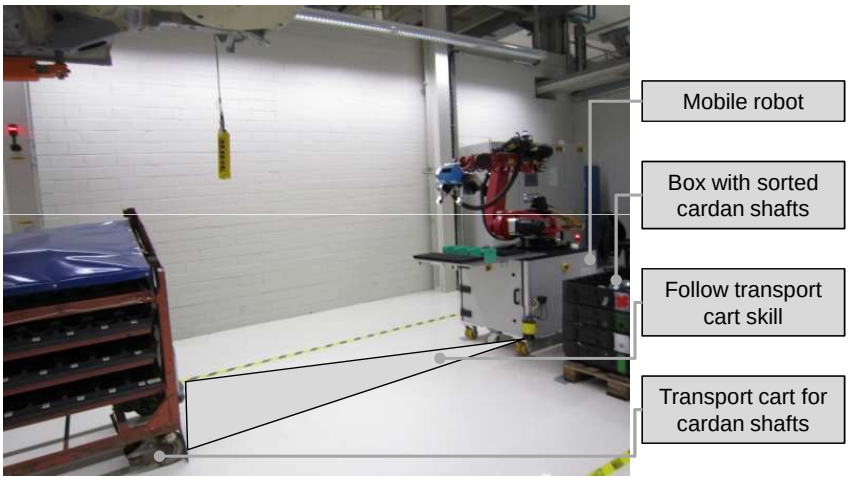


Figure 6.5: Setup for the Validity Check execution in the laboratory.

By using the test environment as presented in figure 6.5, the reconfiguration mechanism can be terminated through the validation of the Composite Skill Agent $C-SA_{follow}$ in real-world. As soon as the sniffed messages are matched with the provided schema, the Skill Agent is integrated permanently in Standard Holon. Table 6.1 gives an overview of the criteria for the successful use case execution and the corresponding results of the *Follow transport cart*-skill.

The specification of the use case provides a basis for both the explanation of implementation details and for a subset of evaluation criteria given in section 7.2 such as the maintenance of productivity or system stability. After the introduction of the use case *Fol-*

Description of criteria for a successful reconfiguration	Use case results
New Skill Description is integrated into the Reconfiguration Holon.	Yes
Used Skills are extracted from the New Skill Description.	SAmove
Execution Agents are created including the knowledge of the Cloned Skill Agents.	Complete set: I-SAmove, I-SAgrip, I-SAdetect, I-SAdeposit.
Condition mismatches are detected and Condition Requests are initialised.	SAmove: Desired: EnvObject Provided: Location
Returning Matching Reports are analysed and a New Skill Input Data is created.	Two FSMStateDescriptions: 1. State: Cl-SAdetect 2. State: Cl-SAmove
New Skill Input Data is transformed into a Composite Skill Agent C-SAx	C-SAfollow
Validity Check is executed and new Skill is integrated permanently in Standard Holon.	Yes
Functionality of the Composite Skill Agent matches with the New Skill Description.	Robot follows the transport cart.

Table 6.1: Overview of the use case results.

low transport cart the implementations of MobComm architecture and agent framework are described in the following.

6.2 Architecture and Agent Framework

The implementation of the MobComm architecture, as presented in chapter 3, requires an agent platform as a basis for the applied software engineering techniques. The resulting Holonic Multi-Agent-System that is built on this platform, is described in section 6.2.1, the implementation of generic Standard Holon agents is focused on in 6.2.2 while section 6.2.3 details the environment interfaces.

6.2.1 Holonic Multi-Agent-System

For the MobComm implementation, the agent platform JADE [Bellifemine et al., 2007] is combined with the JADE eXtension, Jadex [Braubach et al., 2004], as already realised in approaches such as ADACOR [Leitão and Restivo, 2008] or RIA [Guedemann et al., 2008]. Following the advantages evaluated in the literature review in section 2.3, JADE and its extension Jadex are taken as state-of-the-art agent platforms for the implementation of

agent-based manufacturing systems. An implementation overview is given in the literature review in table 2.5 on page 64.

As introduced in section 2.3.1, JADE is a distributed middleware with a flexible infrastructure and an easy extendibility with add-on modules like Jadex. Following [Bellifemine et al., 2007], it is the "most widespread agent-oriented middleware in use today" and complies with the FIPA standards [FIPA, 2005] as explained in section 2.3.2.

Jadex is a hybrid agent architecture for JADE agents that follow the BDI model [Pokahr et al., 2003]. It further supports user-friendly reasoning capabilities by exploiting the BDI model in combination with programming languages like XML or Java. Both JADE and Jadex have been presented and evaluated in the literature review in section 2.3.1. The use of JADE combined with Jadex, as presented in figure 6.6, allows realising behaviour-based agents in Standard Holon and cognitive agents for reconfiguration in Reconfiguration Holon, as introduced in chapter 3.

The JADE Semantic Add-on (JSA) [Louis and Martinez, 2006] was evaluated as an alternative implementation technique for cognitive reconfiguration agents. The JADE Semantic Add-on, however, cannot provide the reconfiguration flexibility as required in Research Task 7 (cf. table 2.8 on page 74) since no ontology support is provided and used concepts have to be handled in pure semantic language formats which hampers system implementation [Bellifemine et al., 2007] [Louis and Martinez, 2006]. In Jadex, a standard agent creation is faster and requires less programming efforts than in JSA which supports the Research Task 8 for a fast adaptability to process changes. Due to this set of advantages and due to its evaluated usage in literature (cf. table 2.5 on page 64), Jadex is chosen for the implementation of the cognitive reconfiguration agents in MobComm.

The presented MobComm implementation is both runnable in simulation environment and in a real-world setup as detailed in section 7.3. Figure 6.6 shows the implementation structure and its environment.

The agents used in Standard Holon, as presented in table 4.3 on page 106 and those utilised in the Reconfiguration Holon, as given in table 4.1 on page 98, run on the same instance of JADE consisting of a set of agent containers, as shown in figure 6.6.

According to the Standard Holon design, a single standard container hosts all Standard Holon agents and executes the real-world manufacturing processes. Reconfiguration is executed in a separate reconfiguration container that is launched optionally in a different machine. All containers and thus all agents share a common agent management system

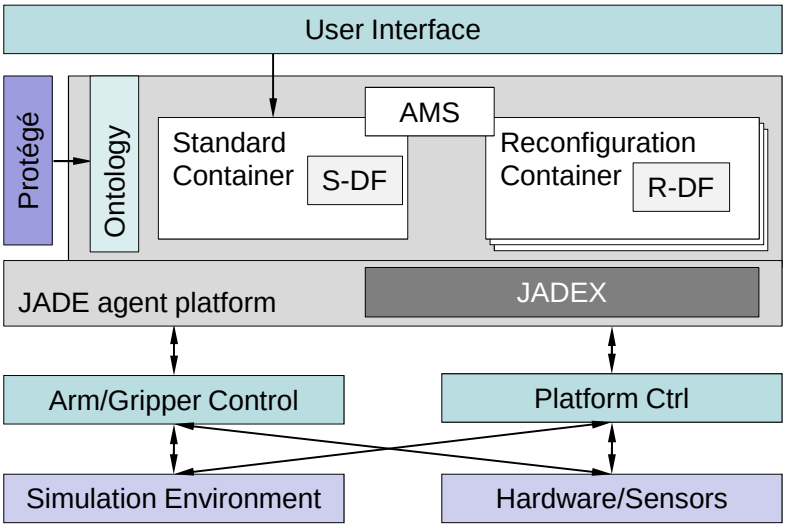


Figure 6.6: Overview of MobComm implementation structure.

(AMS) and a common MobComm ontology as proposed in figure 3.11 on page 91.

For the persistent management of the services offered by an agent, every container has its own Directory Facilitator (DF) that constitutes the *yellow pages* of a holon. The interface between the MobComm implementation and the environment, as presented in figure 6.6, comprises the user interface, the MobComm ontology, and the connection to the hardware control of the robot. The interfaces are further described in section 6.2.3.

The behaviour-based agents in Standard Holon are implemented by exploiting the supplied software engineering techniques of JADE. Listing 6.1 gives an example skeleton of a pre-coded Process, Task, Skill, or Resource Agent in Standard Holon of the MobComm architecture.

```
1  public class AgentName extends MASBaseAgent{
2
3  //Allocate service, agent type, ontology, language and codec to the agent
4  Public AgentName() {
5  super(MASServices.AGENTSERVICE); this.setSkillAgent();
6  getContentManager().registerLanguage (new SLCodec());
7  getContentManager().registerOntology(SHOntology.getInstance());
8  }
9  private HandleRequest handleRequest;
10 //add the request handler for the FIPA REQUEST interaction protocol.
11 @Override
12 protected void addBehaviours() {
13 addBehaviour(new HandleRequest(this));
14 }
```

Listing 6.1: Code extract of the skeleton of Standard Holon Agent.

As presented in listing 6.1, the constructor of every JADE agent registers the used ontology, language, and codec as a basis for upcoming agent interaction. By following the service-based and protocol-specific interaction, as specified in table 4.3 on page 106,

every agent implements a *HandleRequest*-behaviour (cf. appendix listing A.1) to be able to react to incoming *Request*-messages according to the FIPA Request protocol.

The accepted agent interaction in Standard Holon is represented by exchanged messages in figure 6.7 and implements the FIPA Request protocol. Task and Skill Agents are able to exchange *Request*-, *Agree*-, and *Inform*-messages in case of a complied *Request*. *Refuse*- respectively *Failure*-messages are sent in case of a decoding or satisfaction problem (cf. figure 2.33(a) on page 69). If Atomic Skill Agents are used in a standard process, the interaction hierarchy is directed from Process Layer to Task Layer, thereafter to Skill Layer and directly to Resource Layer. Should Composite Skill Agents be involved, the Cloned Skill Agents that are integrated during the reconfiguration process are slotted between Task and Resource Layer, as shown in figure 6.7.

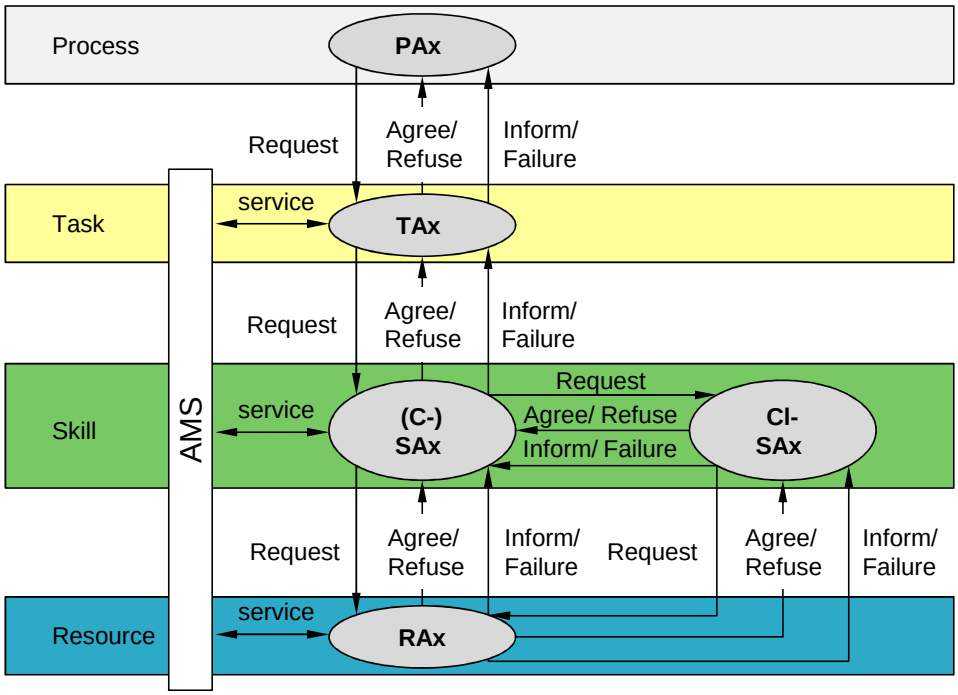


Figure 6.7: Accepted agent interaction in Standard Holon including the service allocation of the Agent Management System (AMS).

Besides the pre-coded Standard Holon Agents presented in listing 6.1, the Generic Skill Agent and the Generic Task Agent are additionally integrated into the presented interaction structure. Both generic agents are detailed in the following.

6.2.2 Generic Standard Holon Agents

The use of Generic Standard Holon agents has a key importance in the proposed reconfiguration mechanism. While the Generic Task Agent (GTA) encapsulates the new process description that has been inserted by the operator, the Generic Skill Agent (GSA) facilitates the on-line integration of a reconfigured Skill Agent into a running process.

The total reconfiguration process is initialised by the instantiation of a Generic Task Agent by the GUI. During the manual insertion of the process change, a Generic Task Agent is created to broadcast the new process requirements to the Reconfiguration Holon. If the reconfiguration mechanism terminates successfully, the Generic Task Agent will be converted into a regular Task Agent, as presented in figure 6.8. Besides the conversion of the Generic Task Agent after reconfiguration, the instant availability of a user-requested skill in Standard Holon causes the registration of the Generic Task Agent as a regular Task Agent. In case the requested skill is not yet available, the reconfiguration mechanism is initialised.

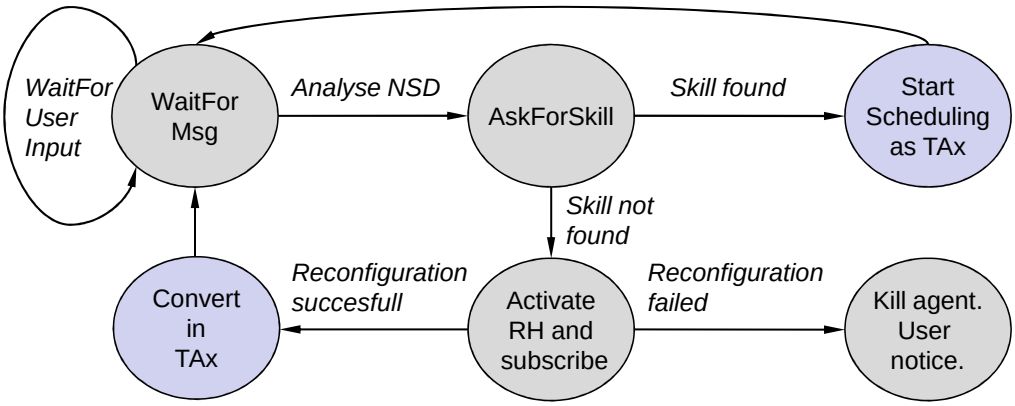


Figure 6.8: The automata of a Generic Task Agent.

While the Generic Task Agent is used during the initialisation phase of a reconfiguration mechanism, the Generic Skill Agent finalises the Distributed Skill Composition and is integrated into Standard Holon as a Composite Skill Agent (C-SA), as presented in figure 6.9. Similar to the conversion of the Generic Task Agent, the Generic Skill Agent transforms into a Composite Skill Agent after successful reconfiguration.

The capability of the Generic Skill Agent to transform specific reconfiguration data into a Composite Skill Agent by using the New Skill Input Data (NSID) is based on its composition of Finite State Machines. As an introduction, the term Finite State Machine

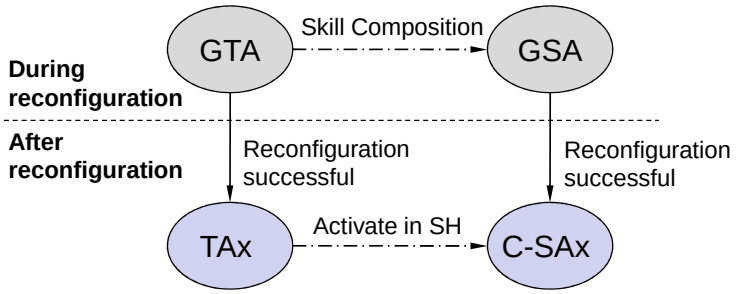


Figure 6.9: Conversions of Generic Task and Generic Skill Agents after successful reconfiguration.

is defined in the following:

Definition 6.1 (Finite State Machine) *An abstract machine that defines a finite set of conditions of existence (called states), a set of behaviours/actions performed in each of those states, and a set of events that cause changes in states (transitions) according to a finite and well-defined rule set [Douglass, 1997].*

The New Skill Input Data that is generated during the Distributed Skill Composition, as proposed in section 4.2, contains a list of *FSMDescriptions*, a list of connecting *Events*, and a HaspMap with the generated parameter allocations. The overview of the New Skill Input Data is presented in figure 4.19 on page 114.

The resulting Finite State Machine of the Generic Skill Agent is given in figure 6.10. The *GSAStates* are divided into a *StartState*, a set of *MiddleStates*, and a finalising *GlobalEndState*. Following section 4.2. where the proposed skill composition is described, condition-based *Events* are provided to connect the *GSAStates* of the Generic Skill Agent. The actual implementation of the Generic Skill Agent integrates only the *LoopSkill*-Event between the first and the last *MiddleState* and ontology-based conditions connecting a *StartState* or *MiddleState* with the *GlobalEndState*.

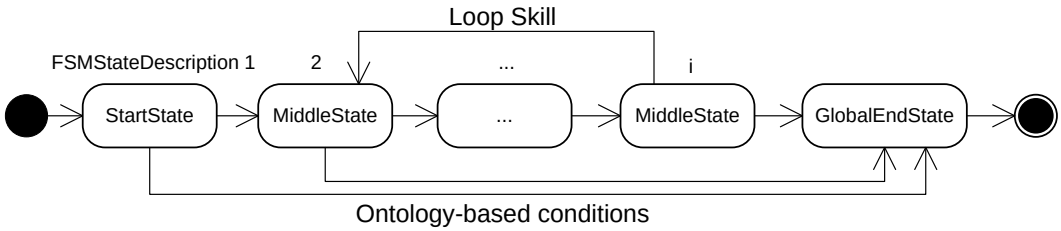


Figure 6.10: Overview of the basic Finite State Machine in the Generic Skill Agent.

The implementation of the presented Finite State Machine exploits the *FSMBehaviour*

that is provided by JADE and enables to register all states and transitions as sub-behaviours with a user-defined scheduling [Bellifemine et al., 2007]. As JADE supports only Finite State Machines at compile-time, a JADE-FSM-Engine is proposed in the work of [Goh et al., 2007]. The flexibility can thus be increased as the engine is able to read FSM-configuration files at runtime [Goh et al., 2007]. The evaluation of the JADE-FSM-Engine and its use for the implementation of MobComm is directed to future work.

The Finite State Machine in figure 6.10 presents all *GSAStates* that are, in turn, structured as inner Finite State Machines with *FSMBehaviours*. The UML-diagram in figure 6.11 shows that the *StartState* is responsible for the integration of the reconfiguration knowledge from the attached New Skill Input Data into the *ExtractParameterAllocations*-state.

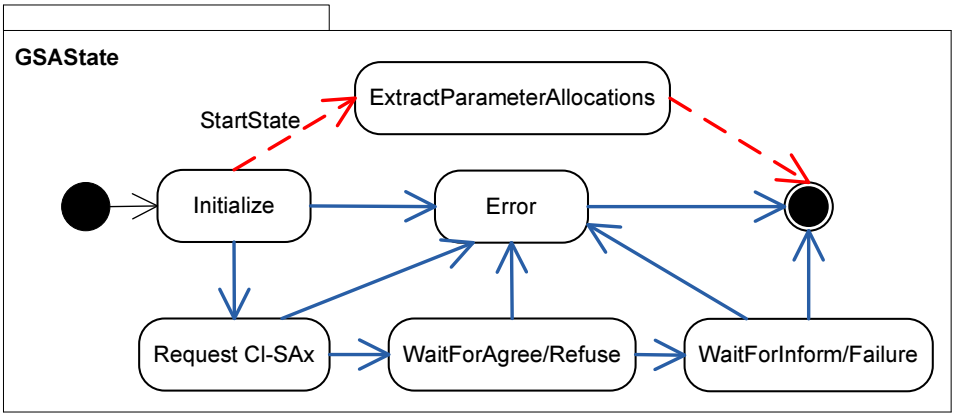


Figure 6.11: Overview of the inner Finite State Machine in a *GSAState*.

In this state, the *handleInputNewSkill* sub-behaviour of the Generic Skill Agent is activated (cf. appendix listing A.2). As described in listing 6.2, the content of a New Skill Input Data is added to the HashMap of the agent and a new *GenericSkillBehaviourFSM* is created for the execution of the *MiddleStates*.

The extraction of parameter allocations by the *StartState* has key importance for the execution of the Composite Skill Agent in Standard Holon. As presented in figure 6.12, the extracted knowledge is stored in a global *HashMap* that is accessible for all *GSAStates* depending on the requirements of the individual states.

The instances of the listed data type have to be forwarded by the corresponding *Request*- or *Inform*-messages and added to the data types in the *HashMap* (cf. figure 6.7). The *MiddleStates*, however, are, after the data storage, able to retrieve the required pa-

```
1  //This behaviour waits until a New Skill Input Data is received.
2    //New Skill Input Data is stored in the agent.
3    NewSkillInputData nsid = (NewSkillInputData)ce;
4    myAgent.setMyIns(nsid);
5    ...
6    //Skill Agent knowledge is extracted from the NSID.
7    myAgent.setMyFsm(nsid.getFsmStateDescriptions());
8    myAgent.setMyName(nsid.getSkillName());
9    myAgent.setDataMappings(masHashMapToHashMap(nsid.getMasHashMap()));
10   myAgent.setMyCond(nsid.getConditionElement());
11
12   //Extract the Event-related States of the Skill Agent.
13   ...
14   for (Iterator iterator =nsid.getAllFsmStateDescriptions();
15         iterator.hasNext();)
16     {fsd = (FSMStateDescription) iterator.next();
17       if (fsd.getConditionFlag())
18       {myAgent.setMyCondState(fsd.getStateName());
19         condition = fsd.getStateName();
20       }}
```

Listing 6.2: Code extract of the *handleInputNewSkill* sub-behaviour of a Generic Skill Agent *StartState*.

rameters including the corresponding instances from the *HashMap* to initialise the *Request Cl-SAx*-state.

Figure 6.12 presents the described retrieval of data to activate a Cloned Skill Agent within *State i*. This includes the extraction of the Preconditions of the *HashMap* and the writing back of the Postconditions in the *HashMap* to allow them to be used by *State i+1* as Preconditions. In the actual implementation the output of *State i* matches the input of *State i+1*. The integration of a dynamic input and output matching system into the Generic Skill Agent implementation is intended to be done in future work.

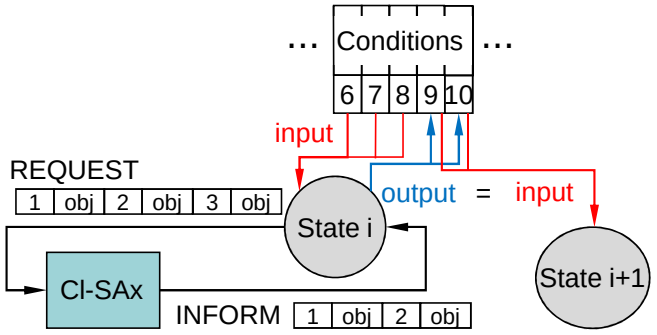


Figure 6.12: Knowledge extraction of the Hash Map for the generation of *GSAStates* in the Generic Skill Agent.

The Generic Skill Agent is thus a nested *FSMBehaviour* that is able to react dynamically to parameters and events resulting from the MobComm reconfiguration.

After both the Generic Task Agent and the Generic Skill Agent have been described in this section, the environment interaction of Standard Holon is focused on in the following before the reconfiguration mechanism is detailed in section 6.3.

6.2.3 Environment Interaction

With reference to the implementation overview in the introduction of this chapter in figure 6.6, the implemented environment interaction can be divided into three interfaces: A user interface, a MobComm ontology, and a hardware interface.

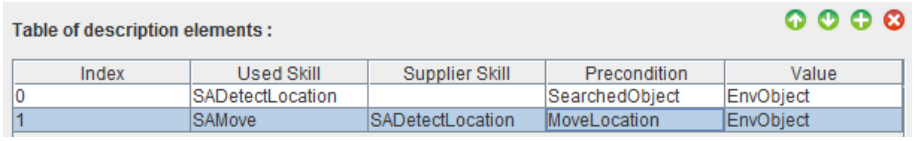
The importance of the user interface for the functional reconfigurability in MobComm is given in the adoption of the process change description. The user interface acts as an input assistance for the process change that is mapped in the New Skill Description. According to the presentation of the internal structure of the New Skill Description, as given in figure 4.5 on page 99, the input lines for the inserted *Reconfiguration Elements* are provided in the graphical interface of figure 6.13(a) and figure 6.13(b). The drop down menu for the Reconfiguration Elements ensures that only known vocabularies are used for the New Skill Description. The immediate verification of a self-consistent New Skill Description by the application of system invariants is part of future work.

The input quality of the Reconfiguration Elements highly affects the level of skill composition during the reconfiguration mechanism as described in section 4.2. The corresponding *Event*-descriptions are inserted by a user interface, as pictured in figure 6.13(c). The actual implementation allows only a basic level of event integration, its advancement is part of future work.

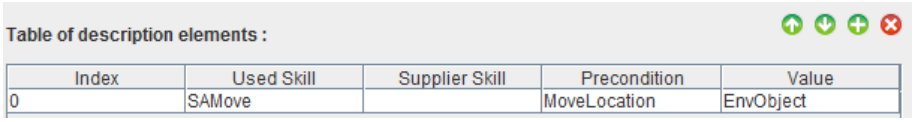
Besides the user-based integration of a process change, a new standard process has to be also activated by the user. This activation, implemented in the user interface as given in figure 6.13(d), contains the assembly of a complete process description including Task Agents, Skill Agents, and Resource Agents.

The semantic connection of the user interface to Standard Holon is provided by an ontology that contains the vocabularies required by a system. The research area of ontologies is introduced in the literature review in section 2.2.3 while the MobComm ontology is presented in figure 3.11 on page 91. The introduced ontology provides shared vocabularies that are used and exchanged by the user interface, Standard Holon, and Reconfiguration Holon.

The implementation of the MobComm ontology is based on a set of JADE classes and realised by the use of Protégé [Stanford Center for Biomedical Informatics Research, 2009]. A screenshot of this tool is provided in figure 6.14 and shows the graphical generation of the *EnvObject* as a subclass of the *ApplicationDescription*. By the subsequent use of the

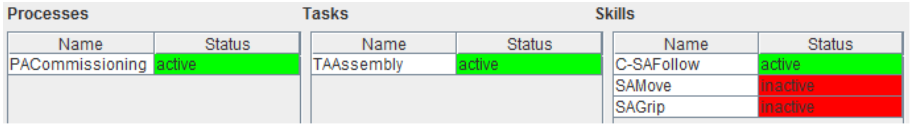


(a) Graphical interface for the insertion of Reconfiguration Elements (Composition Level 1).



(b) Graphical interface for the insertion of Reconfiguration Elements (Composition Level 2).

(c) Graphical interface to insert *Event*-descriptions.



(d) Graphical interface to activate a manufacturing process in Standard Holon.

Figure 6.13: Extract of the user interaction provided in the MobComm Standard Holon.

Ontology Bean Generator, Java beans are created for its user-friendly integration into the JADE agent platform.

The key content of a JADE ontology is its *Concepts* that indicate "existing" entities for agents to "talk and reason about" [Bellifemine et al., 2007]. The content of the operator, environment, and internal part of the MobComm ontology, as given in figure 3.11, is composed of *Concepts* with examples such as the *Application Descriptions Position*, *EnvObject*, or *Location*. Every used *Concept* contains a set of *Fields* added for its specification.

Besides *Concepts*, the MobComm ontology contains *Predicates* that "state the abilities of agents in the form of a Boolean variable" [Bellifemine et al., 2007]. Due to the statement character of a *Predicate*, it is often integrated in the *Inform*-messages used during the reconfiguration mechanism such as a *MatchingResult* that can only be true or false.

The third content of JADE ontologies is the *AgentAction* that "can be performed by some agents" [Bellifemine et al., 2007]. All Atomic Skill Agents, as listed in table 4.3, such as *MoveLocation* or *Deposit*, are added to the MobComm ontology as *AgentActions*.

While *Predicates* and *AgentActions* are used for the internal reasoning in the system,

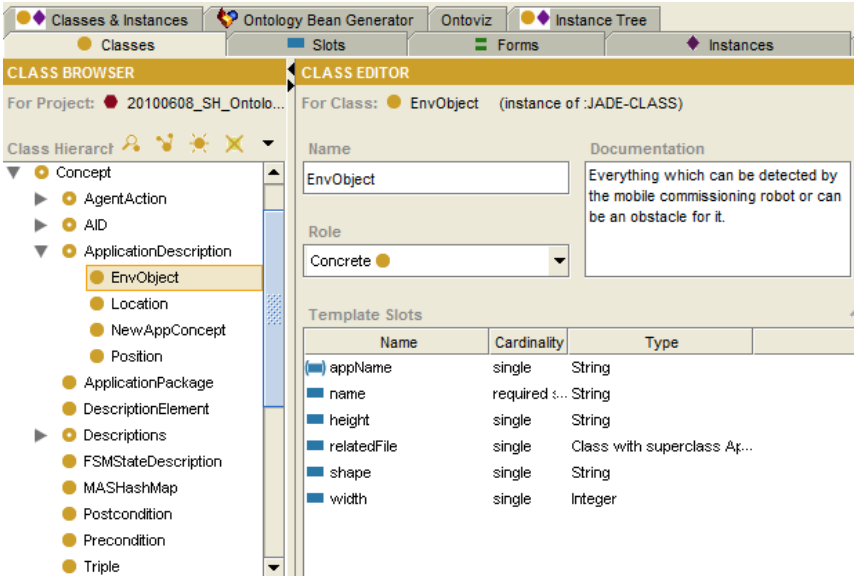


Figure 6.14: Screenshot of the *EnvObject* in the Protégé tool.

the set of *Concepts* in the Environment section of the ontology "represents" the real environment of the robot such as a robot *Position* in space (x,y,z).

To access the abstracted robot hardware in MobComm by Resource Layer, a hardware interface is provided in the environment interaction in MobComm. The hardware interface itself is subdivided into a socket implementation and the integration of the abstracted commands into Resource Agent behaviours. Command implementations at socket level initiate for example the movement of the platform relative to its actual *Location* in the *CmdMoveRelativeData*-class (cf. appendix listing A.3). Hence, this command is used in the *MovePltfToRelCoordinate*-behaviour of the Resource Agent $RA_{platform}$ as shown in the code extract in listing 6.3.

```
1  this.mySocket    = PlatformAgent.socket;
2  ...
3  //Get the actual Location and calculate the difference to target Location
4  double alphaAct = Math.atan2(targetPosY,targetPosX);
5  ...
6  //Send the commands to the Platform control.
7  mySocket.sendCmd(new CmdMoveRelativeData(xDist, yDist,
8          (int)Math.toDegrees(alpha)));
9  mySocket.sendCmd(new CmdPlatStartData());
```

Listing 6.3: Code extract of *MovePltfToRelCoordinate*-behaviour of $RA_{platform}$.

Where a Skill Agent requests $RA_{platform}$ to move relative to its actual *Location* a certain distance (xDist, yDist) in a certain angle (alpha), the *CmdMoveRelativeData*-command is sent to the platform control and initiates the relative movement of the platform in simulation or real-world environment.

The presented implementation follows the proposed hardware abstraction as desired in Task 4 and Task 13 (cf. table 2.8 on page 74) as only the socket implementation or the command structure have to be adapted in case of changed hardware.

After the description of the implemented user interface, the MobComm ontology, and the interface to the robot hardware, the implementation of the reconfiguration mechanism, as suggested in chapter 4 and chapter 5, is detailed in the following.

6.3 Reconfiguration Mechanism

After having described the implementation of the architecture and the applied agent framework in the last section, this section focuses on the reconfiguration mechanism. The agent structure and the interaction mechanism are given in section 6.3.1, while the Generic Skill Composition is covered in section 6.3.2. The Validity Check implementation concludes in section 6.3.3.

6.3.1 Agent Structure and Interaction

Due to the execution of reconfiguration in Reconfiguration Holon, this section focuses on the agent structure and interaction mechanisms used for the reconfiguration agents, as introduced in table 4.1 on page 98.

Since reconfiguration agents are implemented as BDI-agents, the software engineering techniques of Jadex are exploited for the implementation of the reconfiguration mechanism. In general, communication takes place at "two different abstraction levels" [Pokahr et al., 2005] in Jadex.

The intra-agent communication is required to exchange information of different plans inside an agent. In the structure of a MobComm reconfiguration agent, the agent plans are integrated. In a Jadex-agent, several techniques can be used to achieve intra-agent information exchange [Pokahr et al., 2005]. The beliefs of agents are used as triggers for goals and thus implicitly for plans as well. An example belief in the Execution Agents in MobComm is the set of Preconditions and Postconditions of the attached Skill Agent. As presented in section 6.3.2, the Execution Agents execute different plans dependent on the state of these beliefs. The second possibility of intra-agent communication is an internal event such as the*messageEvent* that informs about the arrival of a FIPA *Request*-message.

The inter-agent communication focuses on an information exchange between different

reconfiguration agents, and is based completely on asynchronous *messageEvent* passing [Pokahr et al., 2005]. The *agent definition file* is programmed in XML, as detailed in the appendix in listing A.4, whereas the corresponding plan implementations for the Jadex-agents are written in Java to exploit the advantages of object-oriented programming and the access to third-party libraries [Pokahr et al., 2005].

Due to the *messageEvent*-based interaction in the Reconfiguration Holon, the resulting message exchange, as pictured in figure 6.15, maps the inhomogeneous interaction structure that is adapted to the specific requirements of the suggested reconfiguration mechanism.

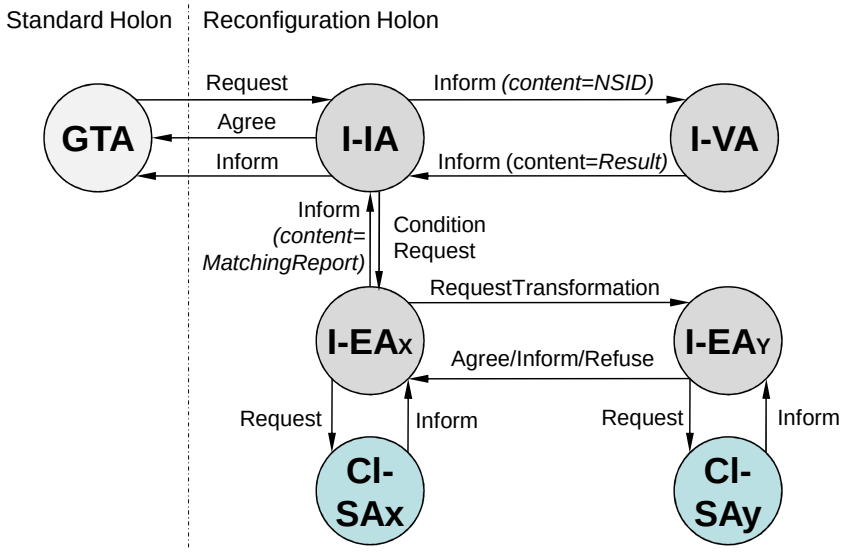


Figure 6.15: Accepted interaction in the Reconfiguration Holon including the interface to the Standard Holon.

While the Generic Task Agent, the initiator of the Reconfiguration Holon and itself a Standard Holon agent, still interacts following the FIPA Request protocol, the interaction between Initiator Agent, Execution Agent, and Validator Agent is based on *Inform*-messages that cause an internal *messageEvent* dependent on the content. Solely interaction among different instances of the Execution Agent requires particular message exchanges as specified in the different composition levels in figure 4.13 on page 109. The Distributed Skill Composition is initialised by a *RequestCondition* of the Initiator Agent and passed among the different Execution Agents with *RequestTransformation*-messages. Both the *RequestCondition*- and the *RequestTransformation*-messages are answered with an *Agree/Inform*-message in case of compliance or by a *Refuse*-message if the condition

cannot be fulfilled by the requested agent.

The goals and plans that are initiated by the described *messageEvents* are focused on in the implementation of the mechanism execution, as presented in the following.

6.3.2 Reconfiguration Mechanism Execution

The reconfiguration mechanism execution is described in the following by the use of goal/plan-trees of the Initiator Agent and the Execution Agents. Goal/plan-trees constitute the correlation between goals, plans, and influencing beliefs of Jadex-agents.

By following the chronology of the reconfiguration mechanism, the goal/plan-tree of the Initiator Agent is presented in figure 6.16 as it is the first agent created in the Reconfiguration Holon. In figure 6.16, the BDI-elements of Jadex agents such as plans, goals, or relevant beliefs are presented additional to the *messageEvents* that trigger the plans or goals. An additional colour code in the goal/plan-trees gives indication about the sender respectively the receiver of the *messageEvent*.

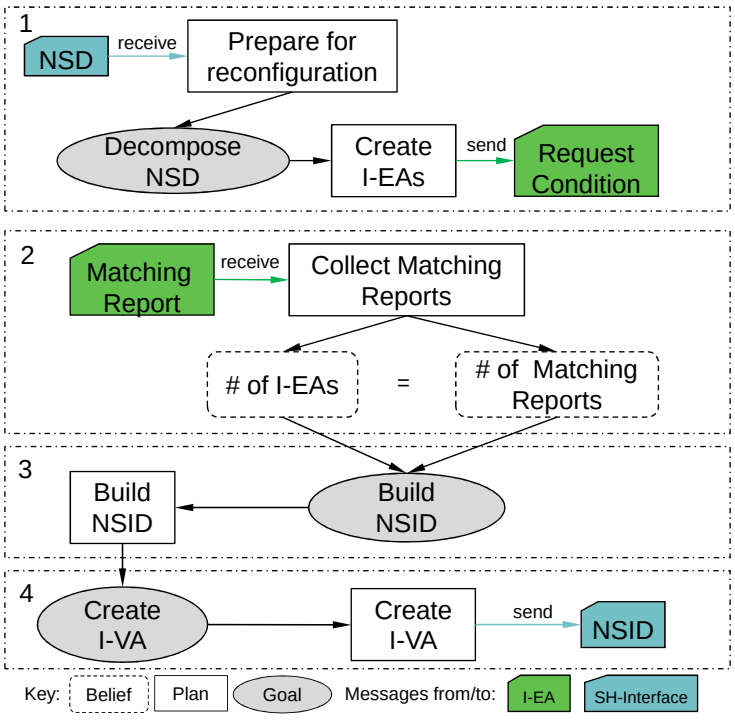


Figure 6.16: Goal/plan-tree of the Initiator Agent including its *messagEvents*.

As detailed in section 4.1, the reconfiguration mechanism is initialised by the New Skill Description-*messageEvent* that is sent by the Generic Task Agent. This *messageEvent* triggers the creation of the required set of Execution Agents (cf. box 1 in figure 6.16) in

```
<p:goals>
  ...
  <!-- Extract NSD, create needed I-EAs and Skill Agent clones -->
  <p:performgoal name="DecomposeNSD"/>

  <!-- Build the New Skill Input Data structure from the collected
       MatchingReports after all reports arrived.-->
  <p:performgoal name="BuildNewSkillInputData">
    <p:creationcondition>
      $beliefbase.numberOfMatchingReports != null &amp;&amp;
      $beliefbase.numberOfEAs != null &amp;&amp;
      $beliefbase.numberOfEAs > 0 &amp;&amp;
      $beliefbase.numberOfMatchingReports == $beliefbase.numberOfEAs
    </p:creationcondition>
  </p:performgoal>

  <!-- Create Validator Agent after the creation of the NSID -->
  <p:performgoal name="CreateValidatorAgent">
    <p:parameter name="input" class="NewSkillInputData"/>
  </p:performgoal>
    ...
</p:goals>
```

Listing 6.4: Code extract of the Initiator Agent goals.

the Reconfiguration Holon.

The second task of the Initiator Agent is the analysis of the collected *Matching Reports* and subsequently, the generation of the New Skill Input Data. Following figure 6.16, the goal *BuildNSID* is activated as soon as all required *Matching Reports* are received by the Initiator Agent. As the reception of the *Matching Reports* marks the successful termination of the Distributed Skill Composition as described in section 4.2, the New Skill Input Data is subsequently created by the Initiator Agent which allows to formalise the reconfiguration output (cf. box 3 in figure 6.16). The agent definition file of the according goal and the depending beliefs are shown in listing 6.4.

The *CreateI-VA*-goal of the Initiator Agent, as presented in the fourth box of figure 6.16, initialises the Validity Check, the implementation of which is elaborated in the next section.

The Execution Agent has a key functionality within the reconfiguration mechanism due to its self-organised execution of the Distributed Skill Composition, as explained in section 4.2. As presented in the goal/plan-tree in figure 6.17, the flexibility of the reconfiguration mechanism is essentially shaped by the capability of the Execution Agent to send and receive *RequestTransformations* and *MatchingResults*.

The initial goal *RequestClone* has the purpose to integrate the agent behaviour of the attached Cloned Skill Agent into the beliefbase of the agent. In order to integrate this knowledge, the Cloned Skill Agent specifically provides Pre- and Postconditions for the Execution Agent which is subsequently able to reason on these conditions as described in section 4.1.2.

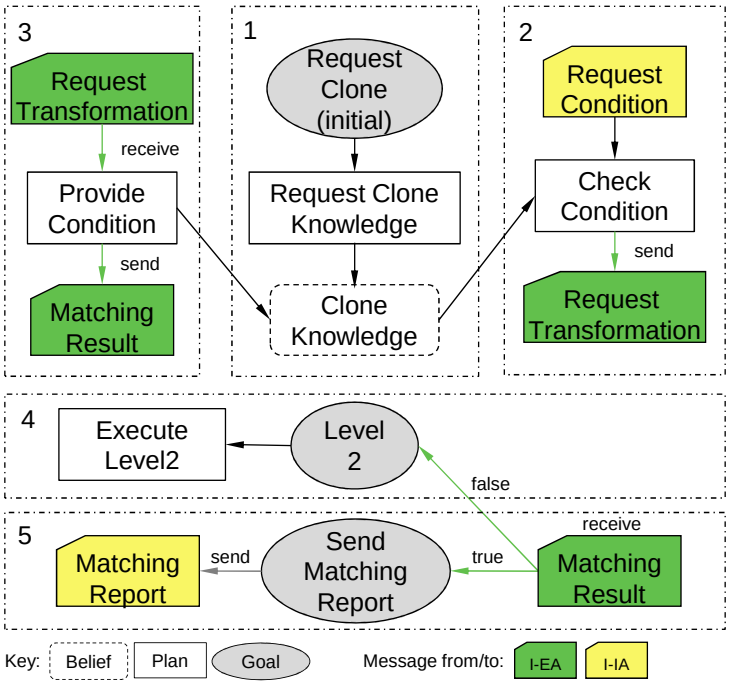


Figure 6.17: Goal/plan-tree of the Execution Agent including its *MessageEvents*.

Once the Cloned Skill Agent knowledge is integrated into the Execution Agent and a *RequestCondition*-message is received, the *CheckCondition*-plan decides whether the condition can be fulfilled by the clone or if a *RequestTransformation* has to be sent to another instance of the Execution Agent (cf. figure 6.17, box 2). The corresponding *CheckCondition*-plan and all further Execution Agent plans are presented in listing 6.5.

In case an Execution Agent in turn receives a *RequestTransformation*-message sent by another Execution Agent, the *ProvideCondition*-plan is activated as presented in box 3 of figure 6.17. In this plan, the beliefset *CloneKnowledge* is checked if the requested condition can be complied with own knowledge. The implementation of the *ProvideCondition*-behaviour contains both the receipt of the *ConditionRequest*-messageEvent and the matching between the *providedClass* respectively the *requiredClass* with the knowledge stored in *CloneKnowledge*.

Depending on the compliance of the *RequestTransformation* that is sent among the Execution Agents, the level of composition is adapted, as described in figure 4.13 on page 109. In the current implementation composition levels one and two are implemented, whereas the third level, as described in section 4.2, is directed to future work.

While the *RequestTransformation*-message is only sent to the supplier skill extracted from the New Skill Description in the first composition level, the second level requires all

```
<p:plans>
  ...
  <!-- This Plan requests knowledge from the attached clone -->
  <p:plan name="RequestCloneActions">
    <p:body class="RequestCloneActionsPlan"></p:body>
    <p:trigger>
      <p:goal ref="requestCloneActions"/>
    </p:trigger>
  </p:plan>

  <!-- Answer to MatchingRequests from I-EAs with MatchingResults -->
  <p:plan name="ProvideConditions">
    <p:body class="ProvideConditionsPlan"></p:body>
      <p:trigger><p:messageevent ref="receiveConditionRequests"/></p:trigger>
  </p:plan>

  <!-- Check if the requested conditions can be fulfilled in Level 1. -->
  <p:plan name="RequestConditions">
    <p:body class="RequestConditionsPlan"></p:body>
      <p:trigger><p:goal ref="searchConditionMatching"/></p:trigger>
  </p:plan>

  <!-- Execute Level 2-->
  <p:plan name="ExecuteLevel2">
  ...
    <p:parameter name="requestedTupel" class="Tupel">
        <p:goalmapping ref="executeLevel2.requestedTupel"/>
    </p:parameter>
    <p:parameter name="ownConditionTriple" class="Triple">
        <p:goalmapping ref="executeLevel2.ownConditionTriple"/>
    </p:parameter>
    <p:body class="ExecuteLevel2"></p:body>
    <p:trigger><p:goal ref="executeLevel2"/></p:trigger>
  </p:plan>
  ...
</p:plans>
```

Listing 6.5: Code extract of the Execution Agent plans.

available Execution Agents and sends the *RequestTransformation*-message to the total set, as proposed in figure 4.13. The implementation of the second level in the agent plan *Level 2* is presented in listing 6.6. After the *ConditionRequest* is processed in the Execution Agent, a *messageEvent* is created to sent the *RequestTransformation* to the remaining set of Execution Agents. In the actual implementation, the search mechanism is terminated after the first condition compliance is received.

```
//Get the requested condition from the NSD and get all I-EAs
MatchingRequest match = (MatchingRequest) getParameter("request").getValue();
...
AgentDescription[] queryResult =
                  (AgentDescription[])ft.getParameterSet("result").getValues();
...
for (int i = 0; i < queryResult.length; i++)
{
  // Create a MessageEvent to send the ConditionRequest to another I-EA
  IMessageEvent me = createMessageEvent("sendConditionRequest");
  me.getParameterSet(SFipa.RECEIVERS).addValue(currentEA);
  me.setContent(match);
  IMessageEvent reply = sendMessageAndWait(me);
  result = (MatchingResult)reply.getContent();

  //Stop if the first result is returned.
  if(result.getResult() == true) {break;}
}
```

Listing 6.6: Code extract of the second composition level in the *Level 2*-plan.

As presented in box 5 of the goal/plan-tree in figure 6.17, the message with the Matching Report that contains the result of the condition matching is the final action of the

Execution Agent. The analysis of the *Matching Reports* including the creation of the New Skill Input Data is subsequently executed by the Initiator Agent, as described above.

The creation of a Validator Agent by the *CreateI-VA*-goal of the Initiator Agent starts the Validity Check, the implementation of which is dealt with in the next section.

6.3.3 Validity Check

The Validity Check, as proposed in chapter 5, is initialised by the Validator Agent in the Reconfiguration Holon but executed by the Process Agent for Validity Check, called VC-PA, in Standard Holon.

The goal/plan-tree of the Validator Agent, as given in figure 6.18, shows that the preparation of the Validity Check is triggered by the receipt of the New Skill Input Data from the Initiator Agent, as described in section 5.2. After the VC-PA is successfully created and the Composite Skill Agent is registered at the Standard Holon-DF, the execution of the Validity Check starts.

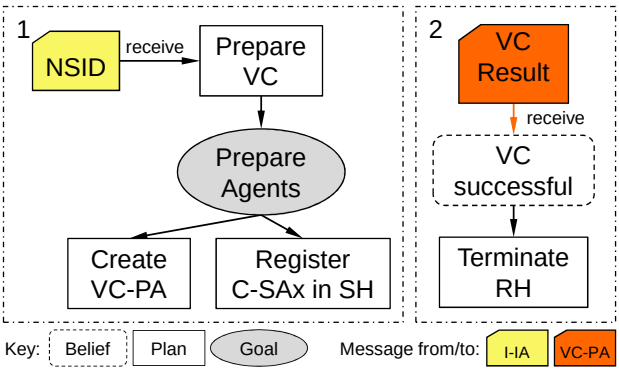


Figure 6.18: Goal/plan-tree of the Validator Agent including its *MessageEvents*.

Further control of the Validity Check execution is within the VC-PA which includes the sniffing mechanism during the execution of the new agent and the subsequent analysis of the sniffed messages. The implementation of the Validity Check is based on the JADE Sniffer that is normally used as a JADE message analysis tool and is integrated in the JADE GUI [Bellifemine et al., 2007]. A specific *VCSniffer*-class is implemented for the execution of the sniffing mechanism. The sniffing schema checks the global and local behaviour and the real-world impact of the Composite Skill Agent, as described in section 5.2. The extraction of the affected agents from the set of available AMS-agents in Standard Holon is presented in listing 6.7.

```
while (agentsinlocation.hasNext()){ //Search all agents in Standard Holon
  OntoAID next = (OntoAID) agentsinlocation.next();

  //Extract all Task Agents for global behaviour check
  if (next.toString().contains("TA")){AID nextAID = next;
  vcSniffer.addSniffedAgents(nextAID);

  //Extract all Cloned Skill Agents for local behaviour check
  }else if (next.toString().contains("Clone")){AID nextAID = next;
  vcSniffer.addSniffedAgents(nextAID);

  //Extract the Composite Skill Agent for local and global behaviour check
  }else if (next.toString().contains(skillName)){AID nextAID = next;
  vcSniffer.addSniffedAgents(nextAID);

  //Extract all Resource Agents and Atomic Skill Agent
    for real-world impact check
  ...
  } else{System.out.println("Unknown agent.");}
}
//Create the VC sniffer with all extracted agents
AgentContainer k = myAgent.getContainerController();
AgentController l = k.createNewAgent("VCSniffer", "VCSnifferAgent",null);
l.start();
```

Listing 6.7: Code extract of the Validity Check sniffing mechanism.

The analysis of the sniffed messages is also executed by the VC-PA, whereas only inconsistent messaging like a missing or additional message from the Composite Skill Agent to the Cloned Skill Agents are recognised in the actual implementation. Both the software level verification with the local and global behaviour as well as the functional level verification with the real-world impact are implemented in the MobComm Validity Check. As the *VCSniffer* is inherited from the classical JADE Sniffer, the execution of the sniffing mechanism is represented in the accustomed JADE sniffer design as presented in figure 6.19.

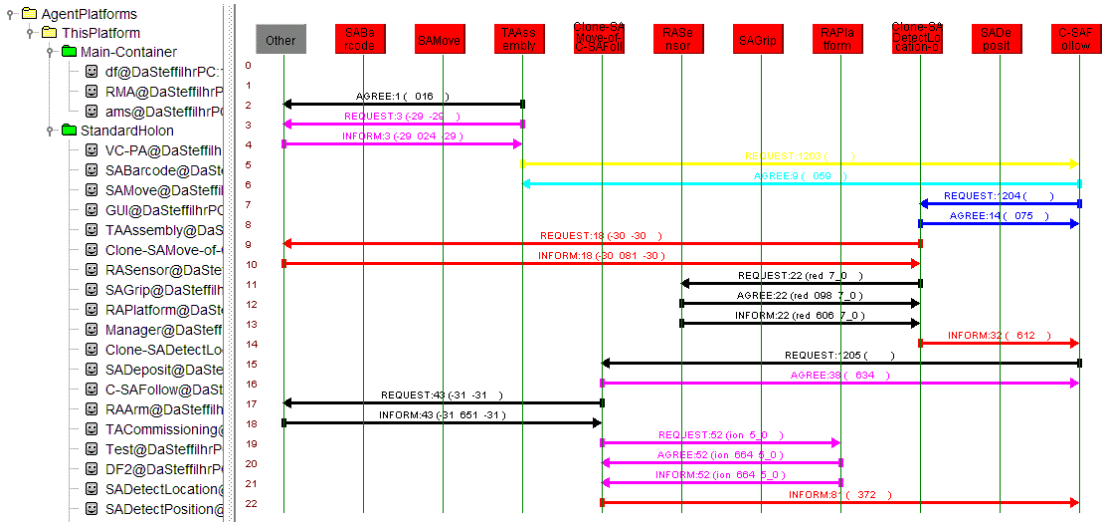


Figure 6.19: Screenshot of the sniffing execution during Validity Check.

The *VCResult* is only returned from the VC-PA to the Validator Agent as true, as de-

scribed in figure 6.18, if the sniffed messages match completely with the prepared message schema in the local and global behaviour as well as in the real-world impact. The receipt of the *VCResult* terminates the reconfiguration mechanism and kills the instance of the Reconfiguration Holon following the specification of the Validator Agent.

6.4 Conclusion

As described in section 6.2 and 6.3, the MobComm implementation provides a basis for conducting the system evaluation in chapter 7. The presented implementation complies with the Supportive Tasks 14/15 (cf. table 2.8 on page 74) as it allows to produce measurable and comparable outcomes for a subsequent evaluation. As the implementation provides the basis to evaluate the key contribution of this thesis, a set of implementation enhancements are beyond the scope of this work and have to be directed to future work.

The event handling during the generation of the Generic Skill Agent and consequently the possibilities to integrate an *Event* have to be extended and optimised. An optimised processing of skill conditions, as described in section 3.3 could rely on the work of [Goh et al., 2007] where the JADE-FSM-Engine is proposed. By the dynamic generation of the complete Finite State Machine instead of a state parametrisation, the Generic Skill Agent structure could be generated more flexibly dependent on the inserted *Event*-descriptions. Listing 6.8 presents a potential way of generating a dynamic Finite State Machine by using the JADE-FSM-Engine.

```
1  Properties fsmProp = new Properties();
2  fsmProp.load(new FileInputStream("FSMAgent.fsm"));
3  FSMBehaviour fsm = new GenFsmBehaviour(this, new DataStore(),
4  fsmFactory.generateFSM());
5  addBehaviour(fsm);
```

Listing 6.8: Code extract of JADE-FSM-Engine. Source: [Goh et al., 2007]

In addition to this enhancement, optimisation strategies within the Validity Check are directed to future work as well. Both the generation of the sniffing schema, as presented in listing 6.7, as well as the processing of the sniffed messages has to be advanced to reach a more robust and reliable validation of the reconfiguration results for an industrial use.

Chapter 7

Experimental Setup and Evaluation Results

The MobComm approach, presented from chapter 3 to chapter 5, is evaluated utilising the system implementation as introduced in chapter 6 and the experimental setup as given in section 7.3 .

The list of research tasks given in table 2.8 on page 74 is used as a basis of this evaluation. Based on the list of research tasks, evaluation metrics form a catalogue in section 7.2 after they have been introduced in section 7.1. The qualitative and quantitative results of these metrics are detailed in section 7.4 after the introduction of the experimental setup in section 7.3.

7.1 Evaluation Methodology

This methodology aims to evaluate the compliance of the suggested work with the given research tasks using an appropriate evaluation catalogue.

In general, the used metrics are classified as measurable or qualitative. While the measurables are assigned to a specific measurement that is indicated with a number in a corresponding unit, the qualitative measurement follows fuzzy rules that result in one of the values of M_{quali}:

$$M_{quali} = \{Low, Medium, High, VeryHigh\} \tag{7.1}$$

For the evaluation of the contribution of this thesis, evaluation metrics are assigned to the individual research tasks. Besides the general classification as qualitative or quantitative, a desired value is set to the metrics for compliance with the set tasks.

Self-organisation (Task 1)

Even if attempts can be found in literature such as [Wright et al., 2001] to quantify self-organisation, a solely qualitative evaluation of which is conducted in this evaluation. Self-organisation is a basic metric as it maps the compliance with Research Task 1. The value of self-organisation is caused by the desire of a low user interaction during the reconfiguration process. As discussed in section 4.4, the level of self-organisation in MobComm has to be balanced with the maintenance of productivity (Task 2) and the dependability of the system (Task 3). Corresponding to that, the desired value of self-organisation in MobComm is set to *High*.

Loss of productivity (Task 2)

The evaluation of the loss of productivity is an elementary metric as self-organisation is combined with manufacturing processes in cycle time. Research Task 2 , however, requires a system that maintains productivity and that does not affect its level during reconfiguration. Corresponding to these requirements that are set by the industrial environment any loss of productivity (i.e. 0%) in MobComm is avoided compared to traditional approaches. The losses caused by hardware failures are excluded by the first and second research assumptions (cf. section 1.3.3).

The measurement of the loss of productivity further provides the main quantitative result of the MobComm evaluation by comparing the proposed reconfiguration mechanism with a state of the art mechanism and the manual execution in terms of production output during functional process changes in industrial commissioning (cf. evaluation results in table 7.15 on page 191).

Predictability of results (Task 3)

The measurement of reconfiguration predictability arises from the required dependability in an industrial environment as mapped in Research Task 3. Even if MobComm does not operate on the safety level of industrial ISO (International Organisation for Standardisation) norms as discussed in section 5.1, the predictability of results has to reach

the measurable value of 100%. Despite and especially because of the high level of self-organisation in an industrial environment, all reconfigured robot functionalities must be predictable for their dependable usage in cycle time.

System stability (Task 4 and Task 5)

The stability of MobComm reconfigurations is a measurable metric for hardware abstraction and configuration independence, as required in Research Task 4 respectively Research Task 5. A stable system is a prerequisite for the qualitative flexibility metric that is additionally assigned to this task. Due to an underlying research implementation, the desired value is 80%. For future usage in industry, this level has to be increased to 99% as specified in table 7.10 on page 181.

Flexibility (Task 4 and Task 5)

The work of [Brennan and Norrie, 2003] that investigates manufacturing flexibility does not give a clear definition of the term flexibility within the manufacturing domain. In view of the wide range of meaning of this term, a definition is provided in the following and adapted from the survey of manufacturing flexibility in [Gupta and Goyal, 1989].

Definition 7.1 (Flexibility) *Flexibility is the ability to vary the steps necessary to complete a task within a defined parts spectrum quickly and adapted to the given environment. Adapted from [Gupta and Goyal, 1989].*

The flexibility metric is supported to give indication about the system behaviour and the surrounding conditions while changing robot configurations and robot hardware. As no learning mechanism or knowledge reuse is integrated into the actual implementation, the flexibility level must be *High*.

Reconfigurability (Task 6)

The level of reconfigurability is a common metric in manufacturing system evaluation as for example in ADACOR [Leitão and Restivo, 2008]. The evaluation of ADACOR, as presented in section 2.2, utilises the reconfigurability metric to measure the support of the system for production load changes.

Even if complex definitions of reconfigurability exist, such as proposed in [Dashchenko, 2006], the specification of this metric in the MobComm evaluation is taken from the

statement in [Leitão, 2004]:

Reconfigurability [...] is the ability to support different manufacturing system configurations [...] with a small customisation effort [Leitão, 2004].

According to this, the reconfigurability metric is able to evaluate the awareness of reconfiguration capabilities. Besides the effort to undertake a reconfiguration, especially the support for different configurations is mapped in this metric including its self-awareness regarding inconsistent user inputs.

Scalability (Task 7)

As a broad range of process changes is required for MobComm in Research Task 7, the system must provide the corresponding scalability. Regarding the dynamics in a productive environment, system scalability is the basis for an enduring operation. Scalability always depends on the environment conditions and follows the question: "How well does it scale and under which conditions? [Rudin, 1997]".

To set a desired value for scalability, the type of metric has to be considered first. Following [Rudin, 1997], scalability is not a binary variable of *scalable* and *not scalable*, instead, it is "a measurable continuum" [Rudin, 1997]. Defining 100% as the linear scalability, means that a value below one corresponds to a good scalability as desired in this evaluation.

Process requirement fulfilment (Task 7)

The process requirement fulfilment results from the required openness for a broad range of functional changes in Research Task 7. A fulfilment of process requirements, however, is strongly related to the functional correctness of the resulting Composite Skill Agent as analysed in section 5.1. This metric provides information about the functionality resulting from a reconfiguration with regard to the inserted New Skill Description. Even if and just as there is no possibility for an on-line verification during reconfiguration, as discussed in section 5.1, this metric is required to result in a *Very High* value.

Adaptability (Task 8)

Adaptability is allocated to Research Task 8 that requires a fast adaptability to new processes. The adaptability to process changes after the insertion of the New Skill Description through the graphical interface is divided into two aspects. The first measurement is the

adaptation time of the new functionality with an excluded Validity Check (t_a) and a maximum value of 60 sec. The total time of reconfiguration, including Validity Check execution (t_{recon}), is set to 900 sec. The value t_a is based on the cycle time in car manufacturing of around 80 sec. The total reconfiguration time, however, follows the pause time of assembly workers of 20 min that can be used for the real-world validation of the Composite Skill Agent.

While the adaptability finalises the list of metrics that are assigned for the validation of the research tasks, the total set is summarised in table 7.1.

Metric	Desired value	Task description		Quantitative	Qualitative
Self-organisation (so)	High	Task 1	Provide a reconfiguration mechanism that realises self-organisation.		x
Loss of productivity (lp)	0	Task 2	Provide a reconfiguration mechanism that does not affect the level of productivity during reconfiguration.	x	
Predictability of results (pr)	1	Task 3	Provide mechanisms that ensure dependability in the use of new functionalities.	x	
System stability (st)	0,8	Task 4	Provide a reconfiguration mechanism that allows hardware abstraction.	x	
Flexibility (fl)	High	Task 5	Provide a reconfiguration mechanism that is robot configuration independent.		
Reconfigurability (rf)	High	Task 6	Provide a reconfiguration mechanism that is aware of the limitations of its reconfiguration capabilities.		x
Scalability (sc)	<1	Task 7	Provide a reconfiguration mechanism that is open for a broad range of functional process changes.	x	
Process requirement fulfilment (prf)	Very High				
Adaptability (ad)	t_a < 60 sec t_{recon} < 900 sec	Task 8	Provide a satisfactory fast adaptability to new processes.	x	

Table 7.1: The set of evaluation metrics and their required values for task validation.

For the evaluation a catalogue and a corresponding framework are required, as detailed in the next section.

7.2 Evaluation Catalogue and Framework

After the research tasks have been assigned to evaluation metrics in section 7.1, methods of measurement are described in the following evaluation catalogue. In addition to this catalogue, the framework that is required by the measurements is defined.

Following a general Multi-Agent System evaluation methodology as presented in [Dimou et al., 2007], the definition of measurements has to answer the question of how the experimental evaluation is supposed to be performed. Nine different ways of measurement are presented in [Dimou et al., 2007], while three of them are selected in this evaluation:

- **Experiment:** An investigation of the **quantitative impact** of methods organised as a formal experiment.
- **Benchmark:** A process of running a number of standard tests using alternative tools/methods and assessing the **relative performance** of the tools against those tests.
- **Qualitative effect analysis:** A **subjective assessment** of the **quantitative effect** of methods and tools, based on expert opinion.

While an experiment focuses on the quantitative impact of the measurement, a benchmark is indispensable if alternatives have to be compared. The qualitative effect analysis, however, is a subjective assessment by an expert and is additionally based on quantitative measurements. In the MobComm evaluation the expert assessment is conducted by an electical engineer, a computer scientist, and the author of the thesis.

In the following, the evaluation metrics of table 7.1 are assigned to one of the presented types of measurements. While the qualitative metrics are elaborated in section 7.2.2, the quantitative metrics are described in the following.

7.2.1 Quantitative Metrics

The quantitative metrics include the measurement of loss of productivity, of predictability of results, of system stability, of scalability, and of adaptability. Their execution parameters are described in the following.

Adaptability (Experiment)

The adaptability is measured by the execution of an experiment and results in two time values: The adaptation time t_a for the provision of the new Composite Skill Agent without the execution of the Validity Check, and the total reconfiguration time t_{recon} until the permanent integration of the new skill into Standard Holon. This differentiation is introduced as the Validity Check is highly dependent on the environment parameters, such as the distance to the end point. The adaptation time t_a is determined by the self-organised reconfiguration mechanism, as introduced in chapter 4.

The corresponding experiment uses the List of Scenarios as introduced in table 7.2. Besides the two variations of the *Follow*-scenario that has already been presented in the use case in section 6.1 and the conditioned *FollowUntil*, the *TrackedGrip*- and *AttachTo*-scenarios are executed for this experiment.

Skill Name	Linguistic description	New Skill Description (NSD)			
		UsedSkill	Precondition	Value	Event
Follow 1	Detect a certain *EnvObject* and follow this *EnvObject* in a Loop.	Move	MoveLocation	EnvObject	Loop
Follow 2		DetectLocation	SearchedObject	EnvObject	Loop
		Move	MoveLocation	EnvObject	
Follow Until	Follow a specific *EnvObject* until the *Location* of the robot is less than 2 meters.	Move	SearchedObject	EnvObject	IfElse
		DetectLocation	MoveLocation	EnvObject	
		If (Location > 2) LOOP else STOP			
Tracked Gripping	Search for *EnvObject* and guide to the *Position.*	Pick	PickPosition	EnvObject	Stop
AttachTo	Attach a gripped Object to a specific EnvObject.	Deposit	DepositPosition	EnvObject	Stop

Table 7.2: List of Scenarios for the measurement of the adaptability values t_a and t_{recon}.

Even if this experiment is executed in the real world for the most part, selected Validity Checks are executed in the simulation environment due to an incomplete environment in the real-world.

System stability (Experiment)

To evaluate two aspects of system stability, a set of experiments is performed. While the

first aspect focuses on the stability of MobComm regarding its independence on a feasible New Skill Description, the second part concentrates on the stability after agent deaths during standard execution and reconfiguration.

As the List of Impossible Scenarios contains erroneous or inconsistent New Skill Descriptions, the system stability after their integration in the Reconfiguration Holon is evaluated in the first experiment. Besides inconsistent modifications of the *FollowUntil*-scenario, named *Err1* and *Err 2*, further deficient scenarios without relation to the List of Scenarios are introduced in table 7.3.

Skill Name	Linguistic description	New Skill Description (NSD)			
		UsedSkill	Precondition	Value	Event
Err1	Searched Object is not a Precondition of Move.	Move	***SearchedObject***	EnvObject	IfElse
		If (Location>5) LOOP else STOP			
Err2	Event cannot be complied as no Position is queried.	Move	MoveLocation	EnvObject	IfElse
		If (***Position***>5) LOOP else STOP			
Err3	Composition Level 3 is not provided.	**DetectPosition**	SearchedObject	EnvObject	Stop
		Move	***MoveLocation***	EnvObject	
Err4	No reconfiguration required. Skill Move.	Move	***MoveLocation***	***Location***	Stop
Err5	GripPosition is not a Precondition of Move.	***Move***	***GripPosition***	Position	Stop
Err6	Usage of two Skill Move cannot be handled.	***Move***	MoveLocation	EnvObject	Stop
		Move	MoveLocation	Location	

Table 7.3: The List of Impossible Scenarios for the evaluation of system stability.

In the *Err3*-scenario, the second level of composition with a *ConditionRequest* between a *Position* and a *Location* as introduced in section 4.2 is not able to compose a new skill. In contrast to the *Err4*-scenario that does not require a reconfiguration and is redirected to Task Level for temporal scheduling ($SA_{movePlfm}$), *Err5* covers the inappropriate combination between *UsedSkill* and *Precondition*. *Err6*, however, is not feasible as $SA_{movePlfm}$ is utilised twice in a New Skill Description which cannot be handled. During the execution of the presented list, the outcome of the reconfiguration and its stability are measured.

The second experiment utilises the use case *Follow transport cart*, as presented in section 6.1 to evaluate the effect of killed agents on system stability. Agents are artificially

killed during standard execution and reconfiguration while the system behaviour is traced in simulation. Every executed scenario is scored with 100 points in case of a total stability while an unstable scenario results in zero points.

The total system stability of MobComm results in the following equation:

$$ST\,(n) = \frac{\sum_{i=1}^{n} \left(st_{points}\,(i)\,/100\right)}{n} \tag{7.2}$$

where $st_{points}(i)$ is the score of the single scenario i and n the number of scenarios.

Loss of productivity (Benchmark, Experiment)

Before the measurement of the loss of productivity is specified, the term productivity is discussed regarding its use in this evaluation. Based on the common definition where productivity is the output resulting from the given input

$$Productivity = Output/Input,$$

the output of an automotive assembly is generally measured in cars per hour. Adjusted to the considered commissioning processes as presented in the use case in figure 6.1(a) on page 138, the output is set to the amount of sequenced components per hour in this evaluation.

The first part to measure the loss of productivity is a benchmark that compares the reconfiguration effort of integrating the new robot functionality *Follow transport cart* for different scenarios. The benchmark measures the loss of sequenced parts during the reconfiguration activities for the following possibilities:

1. Manual commissioning.
2. Semi-automated commissioning with mobile robots
   (a) using sequence-programmed processes,
   (b) using the MobComm reconfiguration mechanism.

As the manual commissioning is actually an applied process in the factory of Audi, it is taken as the basis of operation. The loss of productivity for the listed possibilities is compared to the results of reconfiguration within a manual process.

The loss of productivity in percentages results from the comparison of the manual commissioning with MobComm in the following equation:

$$LOP_{man} = -\frac{PR_{man} - \frac{PR_{mobComm}}{ST}}{PR_{man}} \tag{7.3}$$

where PR_{man} is the amount of handled parts for manual commissioning and $PR_{mobComm}$ is the compared value of MobComm.

The corresponding equation results from the comparison between the sequence-programmed mobile robot and MobComm:

$$LOP_{seq} = -\frac{PR_{seq} - \frac{PR_{mobComm}}{ST}}{PR_{seq}} \tag{7.4}$$

where PR_{seq} is the amount of handled parts for sequence-programmed commissioning and $PR_{mobComm}$ is the compared value of MobComm.

System stability of both the manual and the sequence-programmed execution is assumed with 100% as both are well-established industrial processes. The lost parts of MobComm $PR_{mobComm}$ are divided by the system stability (ST). A positive value points out a loss while a raise is given by a negative value according to the definition of the metric. Following equation 7.3, a benchmark-specific factor (PR_{man}) maps the local condition in the use case, whereas a long-term factor is covered with the integration of system stability (ST).

While the benchmark evaluates the loss of productivity regarding the implementation of new functionalities, the second part is covered by an experiment that focuses on the productivity loss in Standard Holon during the execution of the reconfiguration mechanism. For this experiment, executed in simulation environment, the number of Atomic Skill Agents in Standard Holon is increased to enlarge the amount of Execution Agents in Reconfiguration Holon as well. The process execution in Standard Holon is captured without a parallel reconfiguration, during reconfiguration with a small amount and an increased number of Skill Agents. The percentage raise of execution time during reconfiguration is mapped in LOP_{recon}.

The introduced aspects evaluate both the degree of industrialisation that is mapped in LOP_{man} and the maintenance of productivity in Standard Holon during reconfiguration as given in LOP_{recon}. The intersection of both measurements results in the loss of productivity of MobComm LOP_{total} as presented in equation 7.5:

$$LOP_{total} = \frac{LOP_{man} + LOP_{recon}}{2} \tag{7.5}$$

Predictability of results (Experiment)

For the predictability of reconfiguration results, the measurement as conducted in ADACOR [Leitão and Restivo, 2008] is taken as a basis for this evaluation. Following [Leitão, 2004], the predictability can be measured by repeating the same experiment several times and by the subsequent extraction of the standard deviation. Thus, this metric evaluates the ability of MobComm to create a predictable and repeatable outcome for the equal input.

In ADACOR [Leitão and Restivo, 2008], the predictability measurement includes the actual load of the system, the occurrence of disturbances, or the non-linear dynamics of the manufacturing system. Due to the different scope of ADACOR [Leitão and Restivo, 2008] and MobComm, as reviewed in section 2.2, a stochastic environment that is based on disturbances is excluded in this work. Instead of a stochastic environment, the compliance with the reconfiguration expectations is integrated into the resulting formula, as presented in equation 7.6. The binary reconfiguration expectation $re \in [0, 1]$ gives information about the compliance with the expected outcome of an expert user.

Thus, the predictability formula uses the multiplication with the system stability as a long-term factor and the adding up of the coefficient of variations of the reconfiguration time $v(t_{recon}(i))$ ($[0, 1] = 0 \leq v(t_{recon}(i)) \leq 1$) multiplied with the reconfiguration expectations.

The predictability formula for n experiments is given as follows:

$$PR(n) = ST * \frac{\sum_{i=1}^{n}(1 - v(t_{recon}(i)) * re(i))}{n} \tag{7.6}$$

while ST is the corresponding system stability, n is the number of scenarios, re is the reconfiguration expectation, and v the coefficient of variations.

The coefficient of variations, in turn is defined as the quotient of the standard deviation and the average, as given in equation 7.7:

$$v = \frac{\sigma}{\overline{X}} \; with \; 0 \leq v \leq \sqrt{z - 1} \tag{7.7}$$

while σ is the standard derivation and $\overline{X}$ the average value with z repetitions of a scenario.

Scalability (Experiment)

As the scalability is based on Research Task 7 that requires openness for a broad range of functional process changes, the long term use of the system in the industrial environment and a possible adaptation to growing structures in the factory has to be analysed in this metric. Therefore, the number of Skill Agents in Standard Holon is continuously increased in an experiment based on the use case *Follow transport cart.*

In the first part of the simulated experiment, system performance and adaptation time t_a are measured separately in Standard Holon and Reconfiguration Holon during the expansion of Skill Layer.

In the second part the influence of the used composition level on the reconfiguration capabilities is measured dependent on the amount of Execution Agents. System performance and adaptation time t_a are measured in Reconfiguration Holon in this experiment.

After the introduction of the quantitative metrics, the method of measurement for the qualitative metrics is given in the following.

7.2.2 Qualitative Metrics

To provide comparability for qualitative metrics, a shared method of measurement is introduced and applied for the list of metrics, as introduced in table 7.9.

This method is based on the use of fuzzy sets as surveyed in [Zadeh, 1996]. According to that, a membership function is provided for the qualitative measurements according to figure 7.1.

The definition of a membership function, taken from [Paetz, 2002], is the basis for its application in this evaluation:

Definition 7.2 (Membership function) *Let X be a set. A is called a fuzzy set if there exist a corresponding membership function $m : X \rightarrow [0, 1]$ that is defined everywhere on X. The value $m(x)$ is called membership degree of $\mu(X)$. The region in the data space where $m(x) = \alpha$ we call $\alpha - cut$ if we consider the membership function. The corresponding geometric region we call $\alpha - region$. If additionally the whole rule with the conclusion is considered we speak about $\alpha - rules$. In the special case of $\alpha = 1$, i.e. when considering the*

1-cut of a membership function, we speak about core regions respectively core rules [Paetz, 2002].

For MobComm evaluation, the fuzzy variables *Low, Medium, High,* and *Very High* were introduced in equation 7.1 as M_{quali}. Each of these variables, as presented in figure 7.1, is defined on the base variable. Since a fuzzy set A is a collection of ordered pairs $A=(x, \mu(x))$ where the item x belongs to the universe and $\mu(x)$ is its grade of membership in A, it is necessary to define the membership functions for each fuzzy variable [Leitão, 2004]. For the used fuzzy variables, a trapezoidal function type is presented in equation 7.8 to equation 7.11 with varying function parameters.

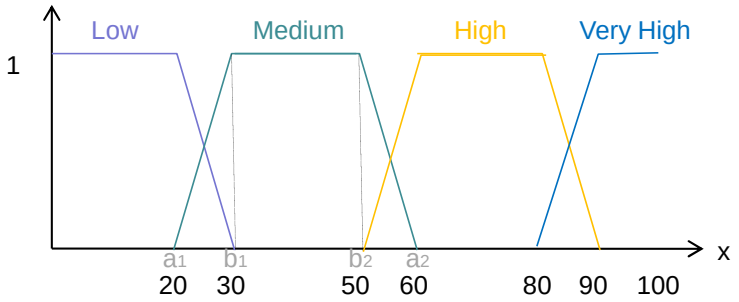


Figure 7.1: Membership function for the qualitative evaluation metrics.

The trapezoid membership function is given with $a_1, a_2, b_1, b_2 \in \Re$ and $b_2 \neq a_2$:

$$\mu_{trapezoid}(x) = \begin{cases} 1 & , \quad x \in [b_1, b_2] \\ \frac{x-a_1}{b_1-a_1} & , \quad x \in [a_1, b_1) \\ \frac{a_2-x}{a_2-b_2} & , \quad x \in (b_2, a_2] \\ 0 & , \quad otherwise \end{cases} \tag{7.8}$$

The fuzzy variable $Medium$ is presented as an example in the following equation:

$$\mu_{medium}(x) = \begin{cases} 1 & , \quad x \in [30, 50] \\ \frac{x-20}{10} & , \quad x \in [20, 30) \\ \frac{60-x}{10} & , \quad x \in (50, 60] \\ 0 & , \quad otherwise \end{cases} \tag{7.9}$$

If $a_1 = b_1$ respectively $a_2 = b_2$, as given for the parameters Low respectively $VeryHigh$, an adapted formula is applied:

$$\mu_{trapezoidCon}(x) = \begin{cases} 1 & , \quad x \in [b_1, b_2] \\ \frac{x-a_1}{b_1-a_1} & , \quad x \in [a_1, b_1) \\ 0 & , \quad otherwise \end{cases} \tag{7.10}$$

The variable $VeryHigh$ is given as an example in the following:

$$\mu_{VeryHigh}(x) = \begin{cases} 1 & , \quad x \in [90, 100] \\ \frac{x-80}{10} & , \quad x \in [80, 90) \\ 0 & , \quad otherwise \end{cases} \tag{7.11}$$

The use of the trapezoidal function is the simplest way to define membership functions without a smoothness surface in the output parameter. The use of a Gaussian function type allows smoothness [Leitão, 2004] but its created complexity would exceed the scope of this evaluation.

Dependent on the used measurements such as experiments and input variables, the qualitative metrics differ in the allocation of fuzzy values that are mapped in individual fuzzy rules.

For the allocation of the metrics self-organisation, process requirement fulfilment, flexibility, and reconfiguration to the fuzzy values, a set of fuzzy rules is described in the following.

Self-organisation (Qualitative effect analysis)

For the measurement of self-organisation, a qualitative effect analysis is conducted utilising the adaptability results after their fuzzification. A qualitative assessment of the complexity of user input for the initialisation of a reconfiguration is integrated into this effect analysis. The allocation of the two input parameters to fuzzy values follows the fuzzy rules that are introduced in table 7.4.

The measurement of self-organisation is based on its definition 2.12 on page 45 [Zadeh, 1963] that contains self-management, structure adaptation, and decentralised control as the main characteristics of applied self-organisation. The decentralised control of MobComm was discussed in section 4.4, while self-management and structure adaptation are mapped into the presented fuzzy rules. The faster a system adapts to a new structure by the application of self-management, the higher its level of self-organisation is. To the contrary,

Adaptability (Fuzzification)	Complexity of user input	**Degree of self-organisation**
-	Very High	**Low**
Low	-	**Low**
Medium	High	**Medium**
High	High	**Medium**
Medium	Medium	**Medium**
High	Medium	**Medium**
Very High	High	**High**
Very High	Medium	**High**
Medium	Low	**High**
High	Low	**Very High**
Very High	Low	**Very High**

Table 7.4: Set of fuzzy rules for measurement of self-organisation.

the complexity of user input has to decline for a raise of the resulting self-organisation.

In case the adaptability is *Low* or the user input is *Very High*, the analysis results in a *Low* self-organisation independent of the value of the remaining parameter. The corresponding values for *Medium, High*, and *Very High* are overviewed in table 7.4.

Process requirement fulfilment (Qualitative effect analysis)

The process requirement fulfilment is a subjective assessment that gives information about the matching of the reconfiguration outcome with the inserted New Skill Description. As presented in the fuzzy rules in table 7.5, the user matching is enhanced with the fuzzification of the predictability as introduced in the previous section.

The matching with the New Skill Description can only adopt the values *Low, Medium,* and *Very High*. While *Very High* corresponds to a full compliance and *Low* outlines the missing fulfilment, *Medium* comes into effect if either the sequence of skills in the Composite Skill Agent or the integrated Event is inappropriate for the inserted New Skill Description.

If the resulting Composite Skill Agent dissents from the inserted New Skill Description, the process requirement fulfilment is set to the value *Low* independent of the predictability value. The total set of fuzzy rules is overviewed in table 7.5.

Predictability of results (Fuzzification)	Matching with inserted NDS (Skill sequence and Events)	**Degree of process requirement fulfilment**
-	Low	**Low**
Low	-	**Low**
Medium	Medium	**Medium**
High	Medium	**Medium**
Very High	Medium	**High**
Medium	Very High	**High**
High	Very High	**Very High**
Very High	Very High	**Very High**

Table 7.5: Fuzzy rules for measurement of the process requirement fulfilment.

Flexibility (Experiment)

The evaluation of flexibility is based on its definition 7.1 in the last section. Flexibility in MobComm reflects the ability of the system to adapt to changing environments in different variations. To evaluate the ability for this adaptation, a List of Changed Hardware is introduced in table 7.6 additionally to the List of Scenarios (cf. table 7.2). A barcode scanner and a corresponding Atomic Skill Agent $SA_{barcode}$ are integrated into the existing robot system for this experiment.

(a) Description of the Atomic Skill Agent $SA_{barcode}$.

Name	Linguistic description	AgentAction	Precondition	Postcondition
$SA_{barcode}$	Read a barcode and provide the according E*nvObject* information	ReadBarcode	Location	EnvObject

(b) Description of the Composite Skill Agents $C-SA_{identify}$ and $C-SA_{check}$.

Name	Linguistic description	New Skill Description (NSD)			
		UsedSkill	Precondition	Value	Event
$C\text{-}SA_{identify}$	Identify the serial number at a car component EnvObject.	Barcode	ReadBarcode	EnvObject	Stop
$C\text{-}SA_{check}$	Check if at a certain Location in the factory a specific EnvObject can be found.	Detect Location	SearchedObject	Location	Stop

Table 7.6: The List of Changed Hardware for the evaluation of flexibility.

The experiment that is executed in the presented simulation environment, evaluates the process requirement fulfilment of the Atomic Skill Agent $SA_{barcode}$, and further for

two Composite Skill Agents $C-SA_{check}$ and $C-SA_{identify}$. While SA_{check} verifies a car component by a barcode label, $SA_{identify}$ focuses on the identification of a component at a certain *Location*.

Self-organisation	Process requirement fulfilment	**Degree of flexibility**
-	Low	**Low**
-	Medium	**Medium**
Low	High	**Medium**
Low	Very High	**Medium**
Medium	High	**High**
Medium	Very High	**High**
High	Medium	**High**
High	High	**High**
High	Very High	**Very High**
Very High	Medium	**High**
Very High	High	**Very High**
Very High	Very High	**Very High**

Table 7.7: Fuzzy rules for the measurement of flexibility.

According to the fuzzy rules in table 7.7, the process fulfilment is combined with the level of self-organisation in this experiment. While the process requirement fulfilment gives indication about the quality of environment adaptation, the self-organisation maps the range of variations that can be applied with respect to a changed environment.

Reconfigurability (Qualitative effect analysis)

Even if the reconfigurability is related to the flexibility that measures the ability to react to changed configurations, reconfigurability additionally describes the effort to change these configurations by following definition 2.2. Flexibility is in fact a subset of reconfigurability as reflected in the fuzzy rules in table 7.8.

According to these rules, the degree of flexibility is combined with an expert-based level of reconfiguration effort. The qualitative effect analysis ends up in two values: First, the hardware-related reconfigurability maps the effort to integrate a new Atomic Skill Agent after hardware changes. Even if the compliance with this type of reconfigurability is beyond the scope of the thesis, it is evaluated in order to being able to define corre-

Degree of flexibility	Reconfiguration effort	**Degree of reconfigurability**
Low	-	**Low**
-	Very High	**Low**
Medium	High	**Medium**
Medium	Medium	**High**
Medium	Low	**High**
High	High	**High**
High	Medium	**High**
High	Low	**Very High**
Very High	High	**High**
Very High	Medium	**High**
Very High	Low	**Very High**

Table 7.8: Fuzzy rules to specify the degree of reconfigurability.

sponding enhancements in future work. Second, the level of this metric for the initial robot configuration gives information about the process-based reconfigurability that can be initiated by the graphical interface, as presented in section 3.4. The reconfigurability metric terminates the specification of the evaluation catalogue, and the total set is summarised in table 7.9.

The required framework resulting from the evaluation catalogue provides the List of Scenarios (cf. table 7.2), the List of Impossible Scenarios (cf. table 7.3), and the List of Changed Hardware (cf. table 7.6) in addition to the use case *Follow transport cart*, as introduced in section 6.1. The experiments or the qualitative effect analysis that are executed in simulation, require specific configurations in this environment, as presented in figure 7.2. As described in the next section, the simulation setup does not allow projecting sensor data. Even if the computational resources of the mobile robot system including the arm/gripper and platform controls are used in simulation, the sensor data has to be modelled for the simulated experiments by former records or random data generations.

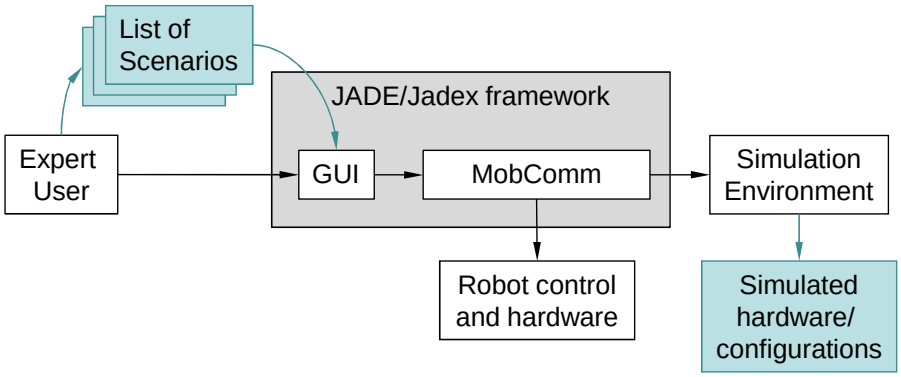


Figure 7.2: Overview of the framework for the MobComm evaluation.

	Metric	Value	Range	Method	Measurement description	Real-world	Simulation
Measurable	Adaptability	t_a < 60sec t_{recon} < 900 sec	$t \in \mathbb{R}^+$	Experiment	Execute **List of Scenarios** and measure t_a and t_{recon} .	x	
	System stability	0,8	$ST \in \mathbb{R}^+$, $0 \leq ST \leq 1$	Experiment	Execute **List of Impossible Scenarios** and trace stability. Execute Use Case and **kill agents.**	x	
	Loss of productivity	0	$LOP_{total} \in \mathbb{R}$, $0 \leq LP \leq 1$	Benchmark	Execute human handling, sequence-programmed and MobComm robot. Measure loss in handled components (LOP_{man}, LOP_{seq}).	x	
				Experiment	Increase **amount of Skill Agents** and measure **loss of productivity** (LOP_{recon}) .		x
	Predictability of results	1	$PR \in \mathbb{R}^+$, $0 \leq PR \leq 1$	Experiment	**Trace reconfiguration expectations** of **List of Scenarios.**	x	
	Scalability	<1	$SC \in \mathbb{R}$, $0 \leq SC \leq 1$	Experiment	**Increase Skill Agents** and measure resources and adaptation time in SH and RH.		x
Qualitative	Self-organisation	High	$AU \in M_{quali}$	Qualitative effect analysis	Evaluate **complexity and amount of user input** and fuzzify **adaptability**. Apply fuzzy rules.	x	
	Process requirement fulfilment	Very High	$PRF \in M_{quali}$	Qualitative effect analysis	Evaluate **matching with NSD** and fuzzify **predictability**. Apply fuzzy rules.	x	
	Flexibility	High	$FL \in M_{quali}$	Experiment	Execute **List of Changed Hardware** and combine with **self-organisation**. Apply fuzzy rules.		x
	Reconfigurability	High	$RC \in M_{quali}$	Qualitative effect analysis	Evaluate **reconfiguration effort** and combine with **flexibility**. Apply fuzzy rules.	x	

Table 7.9: Overview of the evaluation catalogue including the metric specifications and descriptions.

7.3 Experimental Setup

After the evaluation methodology, catalogue, and framework have been described, the experimental setup is introduced in the following. For the comprehensive evaluation of the system both a simulation and real-world environment are provided.

While a simulation environment is generally used to evaluate the system boundaries such as scalability or extendibility, the real-world hardware setup is necessary for the evaluation of industrial requirements such as system stability, maintenance of productivity, or the real-world Validity Check.

Both the simulation and the hardware setup are based on the prototypical mobile robot system manufactured at Audi. An applicable industrial mobile robot has to comply with a certain set of specifications to be perfectly integrated into the given production environment.

The industrial requirements for a mobile robot that is suitable for car manufacturing are overviewed in table 7.10. This set, however, has derived from tests executed in the production environment at Audi and includes both hardware flexibility parameters like gripping geometries and process parameters like availability or energy supply durations [Angerer et al., 2012]. These industrial requirements are only related to the provided hardware of the mobile robot (cf. Hardware Layer in figure 3.5 on page 80) and they are regarded additionally to the research tasks of table 2.8. Their total compliance is beyond the scope of this thesis.

	Industrial requirements	Compliance in test setup
Navigation	Robustness in unstructured environment	Partial
Gripper	Applicability for different part geometries	Yes
Hardware components	Economic components in compliance with industrial standards	Yes
Workload	20 kg	Yes
Workspace	1.8 m	Yes
Availability	99%	No
Energy supply	24 hours	No
Safety	CE labelled application	No

Table 7.10: Industrial specifications of a mobile robot for car manufacturing. Adapted from: [Angerer et al., 2012]

The used robot prototype, however, has to be highly available (99%) with a 24 hours energy supply and a CE-labelled safety system to state EU-safety conformity (Council Decision 93/465/EEC as of 22 July 1993) [1]. Furthermore, a payload of 20 kg and a workspace of 1.8 m [Angerer et al., 2012] has to be supplied by the robot system. These requirements result from the group-wide normed industrial environment for pick-and-place-tasks regarding e.g. heights of shelves, box dimensions, or the average weight of handled parts.

Based on the described requirements, a flexible gripping system for different part geometries and a navigation mechanism for unstructured environment enhances the hardware-related reuseability and transferability of the mobile robot. As the proposed MobComm reconfiguration requires hardware abstraction in Research Task 4, the functional reconfigurability of the total system is restricted by the capabilities of the used robot hardware in the prototype.

The compliance with some requirements that are listed in table 7.10 has not yet been achieved. The 24-hours energy supply, the required level of 99%-availability, and the CE-labelled safety system cannot be provided at this stage of the experimental setup at Audi.

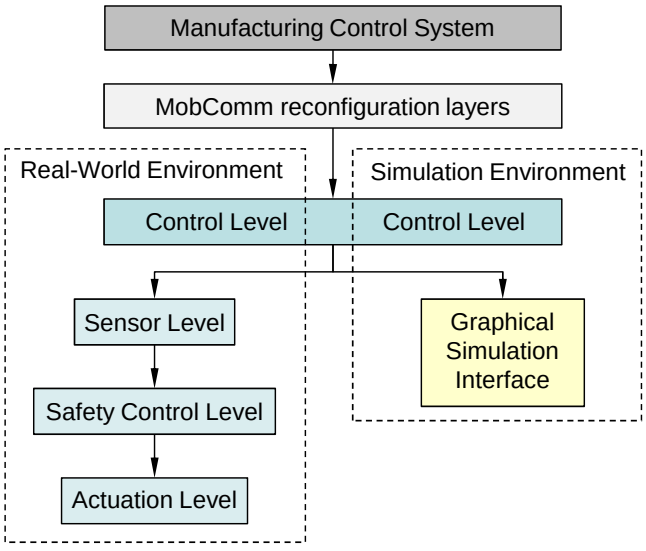


Figure 7.3: Overview of the experimental setup including simulation and real-world environment.

The structure of the robot system used for system evaluation, including software and hardware components, is overviewed in figure 7.3 with a division into four categories: The manufacturing control system, the MobComm reconfiguration layers, the real-world

[1] http://eur-lex.europa.eu/

hardware setup, and the simulation environment.

While the MobComm reconfiguration layers were detailed in chapter 6 including the graphical interface that constitutes the manufacturing control system, the real-world hardware is presented in section 7.3.2 successive to the description of the simulation environment in the following.

7.3.1 Simulation Setup

The simulation environment of MobComm utilises a graphical simulation interface combined with the original hardware control layer of the arm/gripper and the platform as presented in figure 7.4.

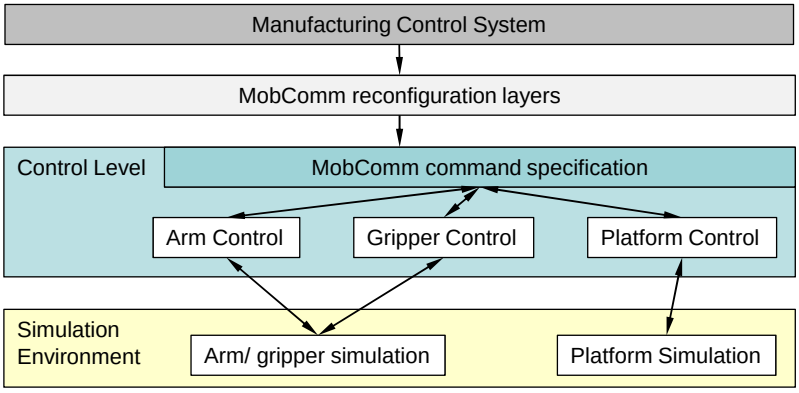
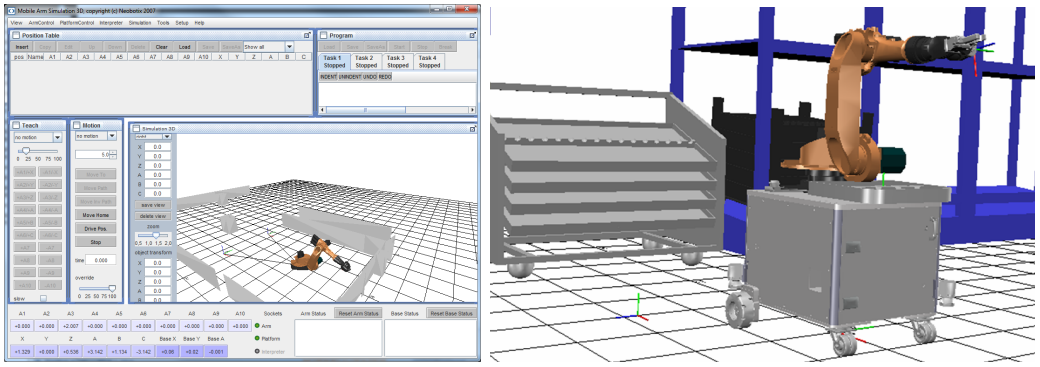


Figure 7.4: Overview of the simulation setup.

General advantages of a simulation setup like a better traceability or an encapsulated test execution with less construction effort than in real-world are also valid for this simulation setup. The MobComm simulation additionally allows to test process executions as the production environment including parts and shelves is recreated in simulation. Figure 7.5(a) pictures a screenshot of the used simulation environment while figure 7.5(b) displays the mobile robot in simulation.

Even if the simulation environment for the mobile prototype excludes sensor information and safety control level such as emergency stop executions, the simulation environment accesses directly the original hardware control and thus provides a suitable tool for a close-to-reality evaluation of a specified set of parameters in section 7.4.



(a) Graphical simulation interface. (b) Mobile robot system in simulation.

Figure 7.5: Screenshots of the graphical simulation interface.

7.3.2 Real-world Setup

The real-world hardware setup of the mobile commissioning system is built at Audi for investigation and development purposes. As overviewed in table 7.10, a payload of 20 kg and a workspace of 1.8 m are realised in the prototype, presented in figure 7.6.

Besides a self-localising platform that is able to navigate in semi-structured environment, a modified industrial manipulator is the main component of the prototype. While the modular gripping system with a passive change station has the purpose to enhance gripping flexibility, the integrated vision system is required to allow a dynamic and exact gripping process. The safety system at hardware level is ensuring basic coverage of safety issues for MobComm applications as introduced in the behaviour analysis in section 5.1.

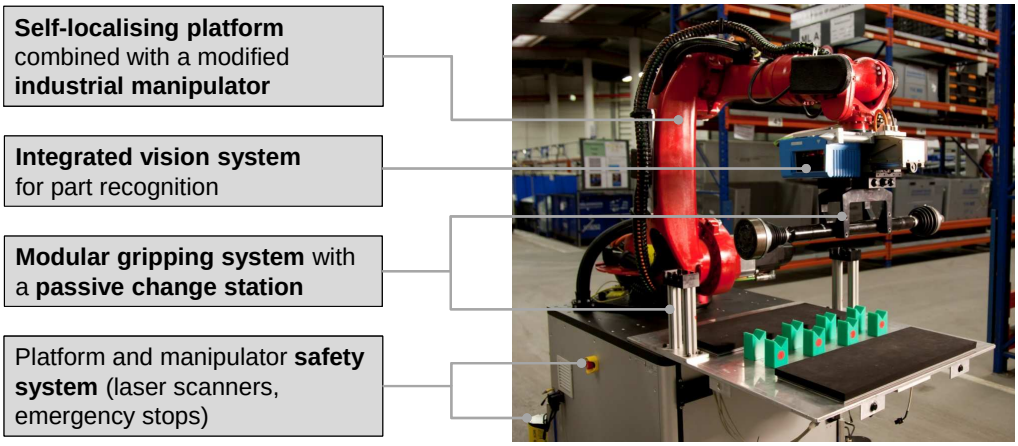


Figure 7.6: Prototype of the mobile commissioning robot.

While the robot arm, the self-localising platform, and the gripper are coupled with their own control entity, as overviewed in figure 7.7, the sensors are linked as sub-parts to

the control level. While the platform requires both laser scanner and platform camera for navigation, the gripper utilises the gripper camera for a guidance of the gripping process. The safety system, however, is authorised to prohibit any hardware access of the complete set of control parts.

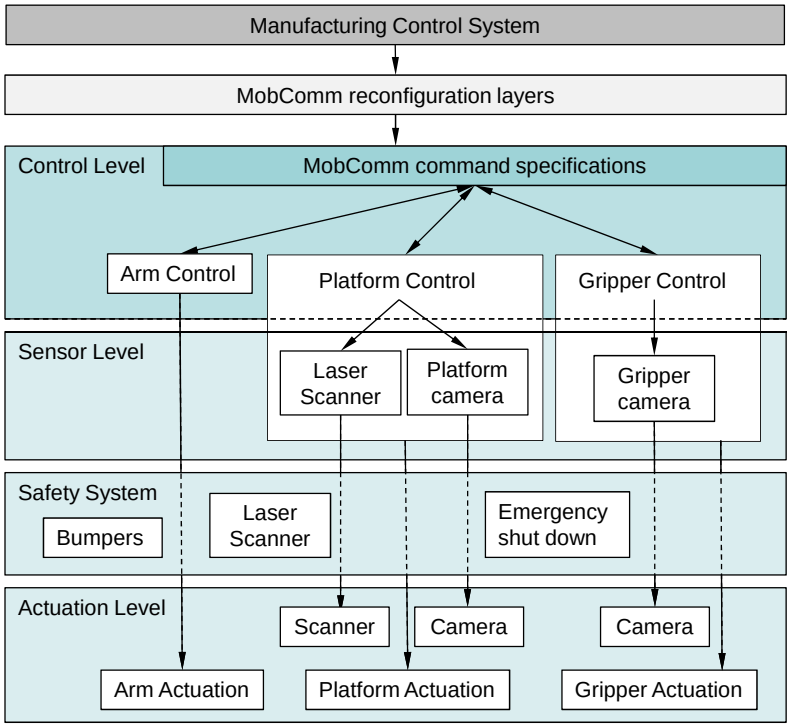


Figure 7.7: Overview of the real-world environment setup.

While the simulation setup lacks only in the missing sensor feedback integration, the real-world setup desires total compliance with the list of industrial requirements as given in table 7.10. This enhancement includes the increase of system availability, an optimised energy supply, and the enhancement of safety for norm conformity.

The system implementation, as introduced in chapter 6, and the presented experimental setup lay basis for the evaluation results in the following and thus for the validation of the research tasks as set in table 2.8 on page 74.

7.4 Evaluation Results

Resulting from the list of research tasks, an evaluation catalogue and the corresponding framework has been developed in the last section. While the qualitative results of the executed experiments, benchmarks, and qualitative effect analysis are given in section 7.4.2,

the quantitative metrics are evaluated in the next section including the ensuing validation of the research tasks.

7.4.1 Quantitative Results

For the set of quantitative metrics that is composed of adaptability, system stability, loss of productivity, predictability of results, and scalability, the evaluation results are detailed in the following.

Adaptability

As described in the evaluation catalogue, adaptability is measured by two time values. While the adaptation time t_a maps the time from the insertion of the New Skill Description until a temporary registration of the Composite Skill Agent in Standard Holon, the reconfiguration time t_{recon} covers the total process until the agent is permanently integrated into the Standard Holon after the Validity Check.

Skill Name	New Skill Description (NSD)				Stable system	Composition level	t_a / t_{recon}
	UsedSkill	Precondition	Value	Event			
Follow 1	Detect Location	SearchedObject	EnvObject	Loop	yes	1	11,24 sec / 62 sec
	Move	MoveLocation	EnvObject				
Follow 2	Move	MoveLocation	EnvObject	Loop	yes	2	13,90 sec / 60 sec
Follow Until	Move	MoveLocation	EnvObject	IfElse	yes	2	14,9 sec / 55,5 sec
	If (Location>5) LOOP else STOP						
Tracked Gripping	Pick	PickPosition	EnvObject	Loop	Yes	2	15,0 sec / 36 sec
AttachTo	Deposit	DepositPosition	EnvObject	Stop	Yes	2	14,6 sec / 37 sec
						Average:	13,9 sec / 50 sec

Table 7.11: Evaluation of adaptability based on the List of Scenarios.

Table 7.11 overviews the results of the executed scenarios including the measured

values for t_a and t_{recon}. The adaptability leads on to an average adaptation time of 13,9 sec ($t_a = 13,9\,sec$) and a mean reconfiguration time of 50 sec ($t_{recon} = 50\,sec$). The experiment parameters are detailed in the appendix in table B.1.

As these results are below the desired values of $t_a < 60\,sec$ and $t_{recon} < 900\,sec$, the corresponding Research Task 8 aiming at a fast adaptability to new processes can be complied by the proposed mechanism.

System stability

By further following the evaluation catalogue, system stability is evaluated as well at two different aspects. The first part traces system behaviour of the List of Scenarios, as presented in table 7.11, regarding the criteria for system stability that are introduced in table 7.12(a).

(a) Criteria for the stability of an evaluation scenario.

	Description of criteria for a stable system
S1	No exception is raised during standard execution or reconfiguration.
S2	No deadlock occurs in the system: Reconfiguration is either terminated or successful.

(b) Criteria for a successful reconfiguration of an evaluation scenario.

	Description of criteria for a successful reconfiguration
R1	New Skill Description is integrated into the Reconfiguration Holon.
R2	Used Skills are extracted from the New Skill Description.
R3	Execution Agents are created including the knowledge of the Cloned Skill Agents.
R4	Condition mismatches are detected and Condition Requests are initialised.
R5	Returning Matching Reports are analysed and a New Skill Input Data is created.
R6	New Skill Input Data is transformed into a Composite Skill Agent C-SAx.
R7	Validity Check is executed and new Skill is integrated permanently into Standard Holon.
R8	Functionality of the Composite Skill Agent matches with the New Skill Description.

Table 7.12: Criteria for stability and reconfiguration success in system evaluation.

Besides the absence of software exceptions, a deadlock-free execution is required for

a positive level of stability. For the complete List of Scenarios a stable system can be demonstrated in the experiment. The details are given in the appendix in table B.2. Further, the List of Impossible Scenarios is executed for this experiment and analysed regarding its stability in addition to the reconfiguration outcome. This input is required as the configuration-independent output is part of the stability evaluation in MobComm.

To evaluate the output of a reconfiguration, a corresponding set of criteria is applied as provided in table 7.12(b). For the set of inserted New Skill Descriptions, criteria R1 to R8 are assessed in this analysis. Only if the total set can be accomplished, a scenario is evaluated as successfully reconfigured. Continuing details of the reconfiguration outcome in the individual scenarios are overviewed in the appendix in table B.4. According to the summarised results in table 7.13, the system remains stable independent of the reconfiguration outcome for the executed List of Impossible Scenarios.

Skill Name	New Skill Description (NSD)				Recon-figuration successful	System stable
	UsedSkill	Precondition	Value	Event		
Err1	Move	***SearchedObject***	EnvObject	IfElse	No	Yes
	If (Location>5) LOOP else STOP					
Err2	Move	MoveLocation	EnvObject	IfElse	Yes	Yes
	If (***Position***>5) LOOP else STOP					
Err3	**Detect Position**	SearchedObject	EnvObject	ANY	No	Yes
	Move	***MoveLocation***	EnvObject			
Err4	Move	***MoveLocation***	***Location***	ANY	No	Yes
Err5	***Move***	***GripPosition***	Position	ANY	No	Yes
Err6	***Move***	MoveLocation	EnvObject	ANY	No	Yes
	Move	MoveLocation	Location			

Table 7.13: Evaluation of system stability for the List of Impossible Scenarios.

The second part of this measurement concentrates on the *Follow transport cart*-use case. The description of a successful use case execution has already been given in table 6.1 in section 6.1. This list of criteria is again assessed after a systematic dispatch of agents. Subsequent to the insertion of the New Skill Description a separate experiment is executed for every scenario with results as given in table 7.14. Both the evaluation of system stability and the dependency on the reconfiguration outcome are measured for every agent crash

(cf. appendix tables B.5, B.6, B.7).

	Killed agent	Reconfiguration successful	System stable	Explanation
Task Agent (SH)	GTA	Yes	Yes	AMS Failure during Validity Check.
Skill Agents (SH)	SA_{move}	No	Yes	Required skill not found. Error handling.
	$SA_{detectLocation}$	No	Yes	No supplier. Error handling.
	SA_{grip}, $SA_{deposit}$, $SA_{detectPosition}$	Yes	Yes	Skill Agents are not required in this use case.
Recon-figuration Agents (RH)	I-EA_{move}	No	No	Requested conditions are not answered or analysed any more. No further actions. Deadlock situation!
	I-$EA_{detectLocation}$	No	No	
	I-$EA_{detectPosition}$, I-EA_{grip}, I-$EA_{deposit}$	Yes	Yes	Composition Level 2: Execution Agents are not required for the Skill composition.
	I-IA	No	No	Missing of the central entity. No further actions. Deadlock situation!
	I-VA	No	Yes	No possibility to execute the VC.

Table 7.14: Stability after killed agents during the execution of the use case *Follow transport cart*.

The death of the initialising Generic Task Agent only produces a handled failure within a successful reconfiguration, similar to the crash of Skill Agents that has no effect on system stability. Whereas the Atomic Skill Agents, that are not involved in the Distributed Skill Composition, do not affect a successful reconfiguration, the participating Skill Agents that have to be cloned by an Execution Agent allow system stability but no positive reconfiguration outcome.

According to system behaviour after the death of unused Standard Holon agents, the death of unnecessary Execution Agents has no impact on the reconfiguration output in contrast to the involved Execution Agents. Due to the implemented inhomogeneous interaction structure, as presented in figure 6.15 on page 154, reconfiguration agents can only proceed if their messages are answered with an *Inform/Agree/Refuse*-message. If the communication partner dies during this interaction, an unwanted deadlock situation occurs. As the discussion of self-organisation in MobComm (cf. section 4.4) showed that the Initiator Agent constitutes a single point of failure in the mechanism, the system cannot cope with the death of the Initiator Agent during reconfiguration and even leads

to a deadlock situation afterwards.

To compensate this undesired behaviour, a mechanism checking the liveliness of agents during their execution, as suggested in [Seebach et al., 2007] has to be investigated for MobComm in future work.

Due to its focus on the initialisation of the Validity Check, the crash of the Validator Agent anticipates a successful reconfiguration but still maintains a stable system.

As the unstable scenarios that are overviewed in table 7.14 only occur with an exceptional death of the agents in a started interaction phase, they are still scored with 20 points while the remaining stable scenarios can be rated with 100 points. Applying the equation 7.2 leads to a stable system in 84% of all executed scenarios ($st(15) = 0,84$). The calculation basis is additionally given in the appendix in table B.8.

The proven independence of the inserted New Skill Description and the partial handling of internal inconsistencies in the holonic Multi-Agent-System are the prerequisite for the evaluation of Research Task 4 and Research Task 5. Even if the evaluation can only be completed with an appropriate level of flexibility, these results provide a solid basis for the validation of these two research tasks.

Loss of productivity

The loss of productivity is evaluated by the application of two measurements. The values LOP_{man} and LOP_{seq} are created by a benchmark where the presented reconfiguration is compared to a reconfiguration with a manual commissioning process and with a sequence-programmed mobile robot. The benchmark results are shown in table 7.15.

For the presented benchmark, the three process variations, as introduced in the evaluation catalogue on page 170, are divided into common process steps:

1. Step: Process change notification: Notification of a system/operator of the process change.

2. Step: Process execution planning: Integration of the changes into the actual process.

3. Step: Worker briefing: Update workers with information about new processes.

4. Step: Hardware adaptations: Implementation of hardware changes, resulting from the second step.

	Human handling	min max	Mobile robot			
			Sequence programming	min max	MobComm	min max
1. Step: Process change notification	Hardcopy notification	1 min 5 min	Manu-facturing control	0,1 min 0,3 min	Manu-facturing control	0,083 min 0,05 min
2. Step: Execution planning	Process planer	30 min 90 min	Process planer	30 min 90 min	MobComm	5 min 15 min
3. Step: Worker briefing	Process planer	10 min 30 min	-	-	-	-
4. Step: Hardware changes	-	-	Maintenance staff	-	Maintenance staff	-
5. Step: Software changes including testing	-	-	Programmer	60 min 150 min	MobComm	0,14 min VC: 1 min 15min
Total time required Minimum, maximum	31 min, 125 min		90 min, 240 min		6 min, 30 min	
Loss of productivity (ø)	Ø 20 min **14 handled parts**		Ø 105 min **73 handled parts**		Ø 8 min **6 handled parts**	

Table 7.15: Results of the benchmark for the loss of productivity measured during the execution of the use case *Follow transport cart*.

5. Step: Software adaptations: Implementation of software changes, resulting from the second step.

The process change notification produces a very small effort with a maximum value of 0,3 min in both robotic applications, whereas the execution planning in the manual scenario is as high as in the sequence-programmed mobile robot. By the application of self-organisation in MobComm the second process step can be reduced to a maximum time consumption of 15 min for the insertion of the New Skill Description. According to that, the software changes for the process change in MobComm are reduced to the average adaptation time of 14 sec ($t_a = 0,14\,min$) and can be executed in parallel to the running process. The average time of 36 sec ($t_{recon} - t_a$) to execute the Validity Check has to be completely added to the loss of productivity as no parallel process execution is possible. The maximum period of time for a Validity Check including waiting times in the real-world is set to 15 min.

Consequently, the calculation of the loss of productivity for the three commissioning possibilities is based on the process steps that cannot be executed in parallel to the

standard process. In table 7.15, these steps are individually marked in grey colour. By averaging these time losses caused by the integration of the functionality *Follow transport cart*, the sequence-programmed robotic application results in 73 lost parts, followed by human handling with 14 lost parts. MobComm shows a loss of 6 handled parts that are solely caused by the execution of the Validity Check in the real-world.

By the application of equation 7.3, the loss of productivity for the benchmark is measured. With a system stability of $st = 0,84$ and 6 lost parts during reconfiguration, it was possible to increase productivity with MobComm by 49% compared to manual commissioning ($LOP_{man} = -0,489$). The raise of productivity can even reach an enhancement of 90,2% ($LOP_{seq} = -0,902$) in comparison with the sequence-programmed mobile robot. For the calculation of the total loss of productivity (LOP_{total}) only the comparison to the manual commissioning (LOP_{man}) is included as this is the actual process execution.

The second aspect of this evaluation has the goal to validate a central contribution of this thesis as described in section 1.3.1. It evaluates the dependency of productivity on the parallel execution of the reconfiguration mechanism in Standard Holon as shown in figure 7.8. While the benchmark provided information about the loss of productivity during the integration of a new functionality compared to other applications, this experiment gives indication of the productivity level during the creation of the new Composite Skill Agent in MobComm. The basis of evaluation is the use case *Follow transport cart*. The Validity Check is not measured during this experiment as its execution requires the termination of the standard process that is traced.

The adaptation time t_a, the time of standard process execution, and the required computational resources are measured. The composition of standard process execution for this experiment is given in the appendix in table B.9. Finally, the experiment consists of three series that include the absence of a reconfiguration, 7 and 200 Execution Agents in Reconfiguration Holon.

The adaptation time and the computational resources are mainly influenced by the complexity of the Distributed Skill Composition, as presented in section 4.2. As the level of composition is determined by the New Skill Description and as a consequence, by the number of Execution Agents in Reconfiguration Holon, the adaptation time increases threefold with the growth of the Execution Agents. The computational resources, however, face an elevation close to exponential. In contrast to these values, standard execution time is effectively linear as presented in figure 7.8. The small variations of this value

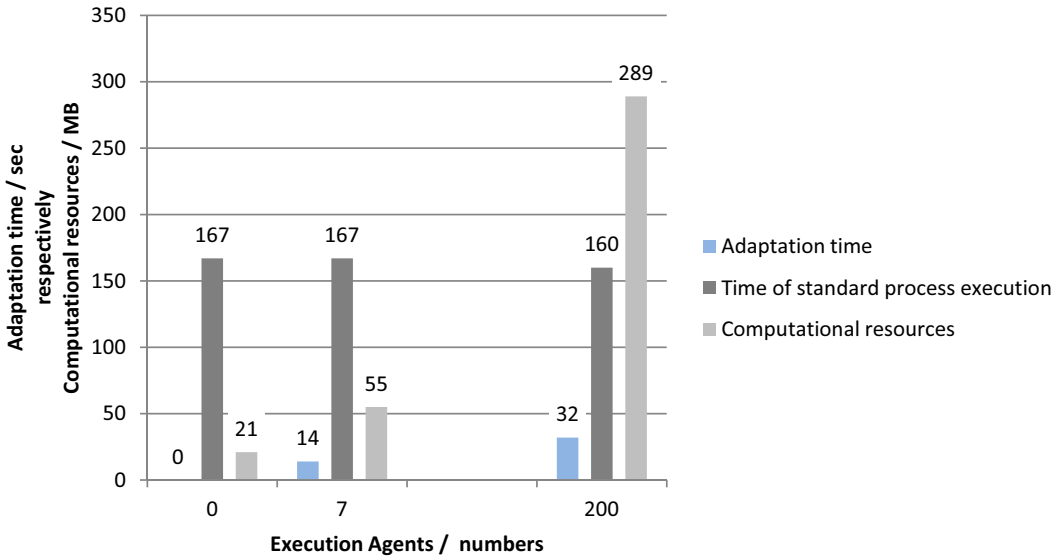


Figure 7.8: Evaluation of the loss of productivity in Standard Holon during the parallel execution of reconfiguration.

have to be ascribed to a divergent platform movement, differing platform angles, and sensor data processing inaccuracies instead of the parallel reconfiguration. As long as the computational resources can be saturated in the used machine, the standard process execution is completely independent of the parallel execution of reconfiguration. Based on the resulting values, the loss of productivity regarding parallel reconfiguration can be designated to a loss of 4,2%:

$$LOP_{recon} = \frac{167 - 160}{167} = 0,042$$

By the application of equation 7.5, the total loss of productivity results in a raise of 22,35%:

$$LOP_{total} = \frac{-0,489 + 0,042}{2} = -0,2235$$

Even though Research Task 2 was evaluated with only a *partial compliance* in the discussion after the Validity Check design in table 5.2 on page 136, these evaluation results demonstrate that the maintenance of productivity can be completely validated for the suggested reconfiguration mechanism.

Predictability of results

After the system's adaptability, stability, and productivity have been evaluated above, the

predictability of results integrates initially a qualitative measurement into a quantitative metric. For the evaluation of the reconfiguration predictability, the expert-inserted reconfiguration expectation is integrated to describe the quality of the reconfiguration outcome as a binary number in equation 7.6. The result of the experiment that is repeated ten times for every scenario of the List of Scenarios and the List of Impossible Scenarios is given in table 7.16.

n	Skill Name	New Skill Description (NSD)				Recon-figuration expectations	V (t_{recon}(n))
		UsedSkill	Precondition	Value	Event		
1	Follow 1	Detect Location	SearchedObject	EnvObject	Loop	1	0,022
		Move	MoveLocation	EnvObject			
2	Follow 2	Move	MoveLocation	EnvObject	Loop	1	0,017
3	Follow Until	Move	MoveLocation	EnvObject	IfElse	1	0,027
		If (Location>2) LOOP else STOP					
4	Tracked Gripping	Pick	PickPosition	EnvObject	Loop	1	0,02
5	AttachTo	Deposit	DepositPosition	EnvObject	Stop	1	0,013
6	Err 1	Move	*SearchedObject*	EnvObject	IfElse	1	0
		If (Location>2) LOOP else STOP					
7	Err 2	Move	MoveLocation	EnvObject	IfElse	0	--
		If (*Position*>2) LOOP else STOP					
8	Err3	Detect Position	SearchedObject	EnvObject	ANY	1	0
		Move	*MoveLocation*	EnvObject			
9	Err4	Move	*MoveLocation*	*Location*	ANY	1	0
10	Err5	*Move*	*GripPosition*	Position	ANY	1	0
11	Err6	*Move*	MoveLocation	EnvObject	ANY	1	0
		Move	MoveLocation	Location			

Table 7.16: Evaluation results for the predictability of results.

By the application of the predictability equation 7.6 of the evaluation catalogue, the coefficient of variations is calculated for the set of 11 scenarios as overviewed in table 7.16 and detailed in the appendix in table B.10. The multiplications of the single values with their reconfiguration expectation allows to exclude the "incorrect" reconfigurations as applied for scenario 7 in table 7.16. The multiplication with the system stability, however, maps the long-term stability into this measurement. The created results lead to

a predictability of 76% of all regarded reconfiguration outputs ($PR(11) = 0,76$).

Similar to the discussion in the reconfiguration chapter in section 4.5 and the Validity Check chapter in section 5.3, these results lead as well to a partial compliance of Research Task 3. The desired dependability of the reconfiguration results cannot be completely fulfilled by this thesis with a value of $pr = 0,76$. According actions to be taken in future work for a total compliance are presented in chapter 8.

Scalability

As the scalability is associated with Research Task 7 requiring the openness for a broad range of functional process changes, a long term use in the industrial environment and a possible adaptation to growing structures in the factory are regarded in this experiment. The number of Skill Agents in Standard Holon is continuously increased while the use case *Follow transport cart* is executed. The first part is an extension to the experiment presented for the evaluation of the loss of productivity in figure 7.8. The adaptation time t_a and the required computational resources are captured during the execution of the standard process *Commission cardan shaft*. As opposed to figure 7.8, there is no parallel execution of reconfiguration in this experiment. The results given in figure 7.9 only give indication about the scalability of Standard Holon.

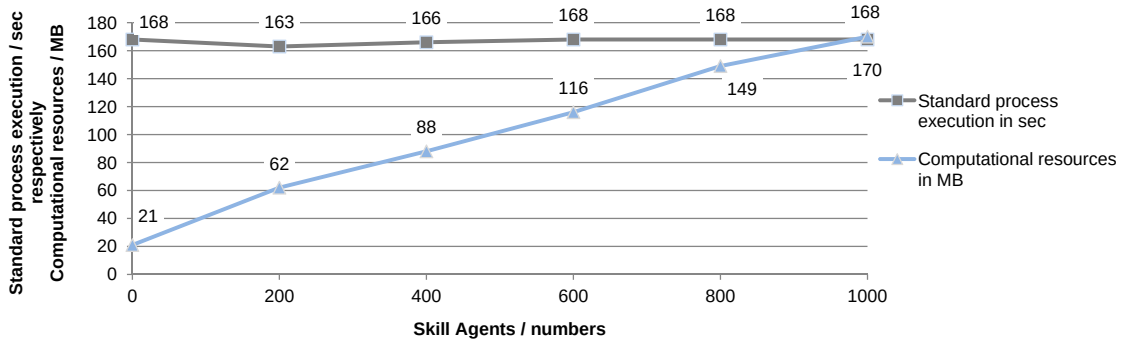


Figure 7.9: Scalability of Standard Holon under consideration of a growing Skill Layer.

In case new Skill Agents are inserted, the required services are requested from the Agent Management System without causing communication overhead in Standard Holon. The service allocation in Standard Holon prevents high message load and thus provides a scalable process execution in Standard Holon, as presented in figure 7.9.

In contrast to the effect of an increasing number of Skill Agents in Standard Holon, an expanded Skill Layer affects scalability of Reconfiguration Holon. Figure 7.10 shows its

dependency on the number of Execution Agents that are created during the Distributed Skill Composition. The scalability is evaluated by measuring the required adaptation time t_a and the computational resources. For a number of Execution Agents between two and 700, both the computational resources and the reconfiguration time grow linearly in Reconfiguration Holon during the execution of the use case *Follow transport cart*. For the experiment whose results are given in figure 7.10, no parallel execution of standard process was regarded.

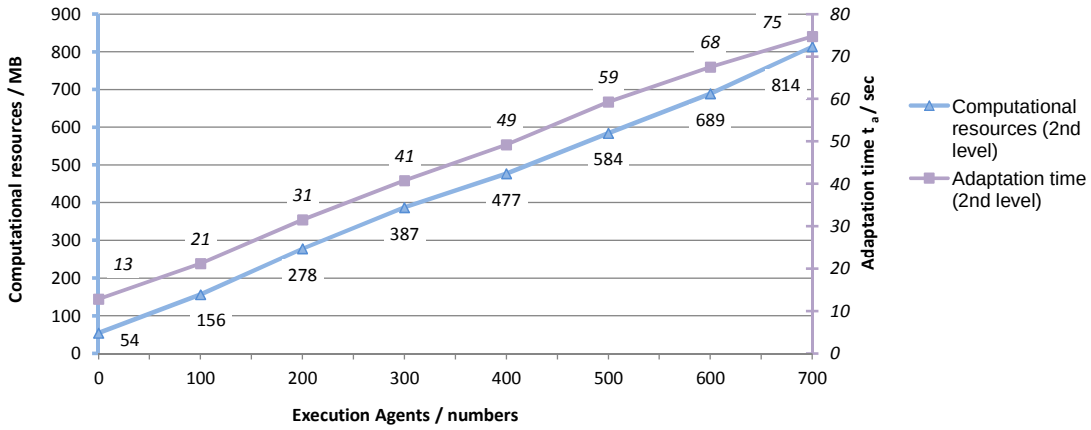


Figure 7.10: Evaluation of scalability in Reconfiguration Holon under consideration of a growing number of Execution Agents.

The experiment has been limited to 700 Skill Agents in Standard Holon and thus to the same amount of Execution Agents in Reconfiguration Holon. As every Skill Agent maps a separate functionality of the connected robot hardware, the chosen amount of skills is a realistic maximum for an industrial mobile robot. Due to the discussion in section 4.4, the Composite Skill Agents are not used for future reconfiguration mechanisms.

The influence of reconfiguration on standard process execution has already been analysed as negligible for the loss of productivity in figure 7.8. As given in figure 7.8, the rise of Skill Agents only affects the adaptation time in the Reconfiguration Holon whereas the standard execution time remains stable until saturation of the computational resources in the used machine is reached.

Standard Holon is scalable independent of Skill Layer complexity in contrast to Reconfiguration Holon that is dependent on the number of Execution Agents. The relationship between the Skill Layer size and the communication load in Reconfiguration Holon is analysed in the following.

The complexity in Reconfiguration Holon is mainly influenced by the execution of the Distributed Skill Composition. Dependent on the quality of the New Skill Description, the applied composition level decides about the arrangement in the Reconfiguration Holon. This dependency is measured in the Reconfiguration Holon by the *RequestTransformation*-messages that are generated as a reaction to the *ConditionRequest* of the Initiator Agent.

In case the first composition level is applicable, the *ConditionRequest* requires only one *RequestTransformation* as the according supplier is already provided in the New Skill Description. The *RequestTransformtion*-messages that arise from one *RequestCondition* rise with a factor $n-1$ in the second level as all Execution Agents are addressed. Resulting in a raise of $f(n) = (n-1)n$, the third composition level increases the message load immensely due to the multi-part reconfiguration algorithm.

As the computational resources of a machine are limited, the scalability of Reconfiguration Holon is reduced especially during the application of the second and third level of composition . Figure 7.11 gives an overview of the three composition levels up to 1000 Execution Agents.

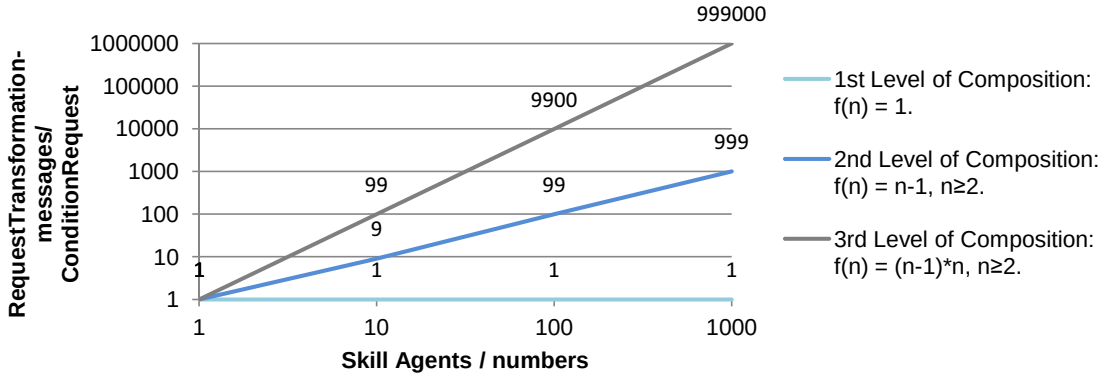


Figure 7.11: Message load per *RequestTransformation* in Reconfiguration Holon under consideration of different composition levels.

Even if Standard Holon provides a very good scalability with a value of one, the Reconfiguration Holon has to be evaluated as insufficiently scalable with the resulting value of $<< 1$.

Based on these results, Research Task 7 requiring the openness for a broad range of functional changes can only be complied partially by the proposed reconfiguration mechanism due to the lack of scalability in Reconfiguration Holon.

In addition to the created results for the adaptation time $t_a = 13,9$ *sec*, the reconfi-

guration time $t_{recon} = 50\ sec$, the system stability $st = 0,84$, a raise of productivity of $lop_{total} = 0,22$, the predictability of results $pr = 0,76$, and the scalability $sc << 1$ terminate the evaluation of the quantitative metrics. By following the evaluation catalogue, as introduced in table 7.1, the qualitative metrics are focused in the following section.

7.4.2 Qualitative Results

While the last section focused on the evaluation of quantitative results, the measurement of the qualitative metrics by application of fuzzy rules will be described in the following.

Self-organisation

The degree of self-organisation in MobComm is based on the fuzzy set, as introduced in table 7.4. The fuzzification of the adaptation time and the complexity of user input for a reconfiguration are required as input variables. The adaptation time is integrated into this evaluation as the efficiency of the self-organised reconfiguration is mainly mapped by this time value. The reconfiguration time t_{recon} is especially influenced by environment conditions and less by the applied self-organisation. The corresponding assignment of the adaptation time to the fuzzy values of M_{quali} is given in table 7.17(a).

(a) Fuzzification of the adaptability metric.

Adaptation time t_a	Value in fuzzy set (adaptability)
< 10 sec	Very High
< 15 sec	High
< 20 sec	Medium
> 20,1 sec	Low

(b) Assignment of the fuzzy value *Complexity of user input.*

Complexity of user input	Value in fuzzy set
Ambiguous and difficult correlations in non-natural language.	Very High
Ambiguous and difficult correlations in natural language.	High
Distinct and difficult correlations in non-natural or natural language.	Medium
Distinct and simple correlations in natural language.	Low

Table 7.17: Assignment of adaptability and complexity of user input to the fuzzy values of M_{quali}.

The classification of the expert-based assessment of input complexity is shown in

table 7.17(b) where ambiguity, difficulty, and the insertion as natural or non-natural language are used as description elements.

During the execution of the List of Scenarios, an expert user evaluates the required input complexity by the application of table 7.17(b) while measuring the adaptation time. Table 7.18 presents the individual scenarios and the resulting level of self-organisation.

Skill Name	New Skill Description (NSD)				t_a	Fuzzification	Complexity	Self-organisation
	UsedSkill	Precondition	Value	Event				
Follow 1	Detect Location	SearchedObject	EnvObject	Loop	11,24 sec	High	Medium	Medium
	Move	MoveLocation	EnvObject					
Follow 2	Move	MoveLocation	EnvObject	Loop	13,90 sec	High	Low	Very High
Follow Until	Move	MoveLocation	EnvObject	IfElse	14,9 sec	High	High	Medium
	If (Location>2) LOOP else STOP							
Tracked Gripping	Pick	PickPosition	EnvObject	Loop	15,0 sec	Medium	Low	High
AttachTo	Deposit	DepositPosition	EnvObject	Stop	14,6 sec	High	Low	Very High

Table 7.18: Evaluation results for self-organisation based on the execution of the List of Scenarios.

The insertion of a single *Reconfiguration Element* and a *Loop-Event* is classed as a *Low* complexity whereas multiple *Reconfiguration Elements* and a complex *Event* description, as given in the scenario *FollowUntil*, causes *High* complexity in the user input. According to these results, self-organisation in MobComm varies from *Medium* to *Very High*. As this assignment results in the average fuzzy value of *High*, the self-organisation as desired in table 7.1, can be complied with the proposed work leading to a complete fulfilment of Research Task 1.

Process requirement fulfilment

To measure the fulfilment of process requirements, a qualitative effect analysis is conducted following the set of fuzzy rules as introduced in table 7.5. The fuzzified predictability of results and an expert-inserted matching with the New Skill Description is required as input variables. The fuzzification of the predictability is given in table 7.19(a) with the corresponding assignment to the fuzzy values M_{quali}. For the creation of the individual predictability values, equation 7.6 is applied for a single scenario $pr(1)$.

The expert-based assessment of the matching with the New Skill Description follows the description given in table 7.19(b). Only three fuzzy values can be assigned, with *Medium* stating either the compliance of the skill sequence or the Event integration in addition to *Very High* for a complete fulfilment and *Low* for no compliance of the New Skill Description.

(a) Assignment of predictability results to fuzzy values.

Predictability of results	Value in fuzzy set
0,91 – 1	Very High
0,66 – 0,9	High
0,41 – 0,65	Medium
0 – 0,4	Low

(b) Assignment of the degree of matching with the New Skill Description to M_{quali}.

Matching with New Skill Description	Value in fuzzy set
Sequence and Event match completely	Very High
Either sequence or Event match	Medium
Neither sequence nor Event match	Low

Table 7.19: Assignment of the input variables to fuzzy variables for the evaluation of the process requirement fulfilment.

For this qualitative effect analysis, an extract of the List of Impossible Scenarios is executed in addition to the List of Scenarios. The two items of the List of Impossible Scenarios are added as they are only inconsistent or impossible in the actual system but not error-prone. As the remaining entries of this list contain error-prone New Skill Descriptions, their evaluation is not integrated into this analysis.

Table 7.20 shows the *High* predictability value for the total List of Scenarios and the *Very High* matching with the New Skill Description. As a consequence, the fulfilment of process requirement also results in a *Very High*-value for the total List of Scenarios. Since the inconsistent New Skill Description of *Err 2* is executed as specified by the system, the *Low* predictability is combined with a *Very High* matching result. This results in a *Low* fulfilment of process requirements by the application of the fuzzy rules introduced in table 7.5. In contrast to the low value of *Err 2*, the consistent New Skill Description in the *Err 3*-scenario provides a *Very High* predictability, whereas the matching with the New Skill Description is *Low* due to the failed reconfiguration. Caused by the missing third composition level in the actual implementation, *Err 3*-scenario also implies a *Low* process

	Skill Name	New Skill Description (NSD)				Fuzzification predictability	Matching with NSD	Process fulfilment
		UsedSkill	Precondition	Value	Event			
1	Follow 1	Detect Location	SearchedObject	EnvObject	Loop	High 0,82	Very High	Very High
		Move	MoveLocation	EnvObject				
2	Follow 2	Move	MoveLocation	EnvObject	Loop	High 0,82	Very High	Very High
3	Follow Until	Move	MoveLocation	EnvObject	IfElse	High 0,82	Very High	Very High
		If (Location>2) LOOP else STOP						
4	Tracked Gripping	Pick	PickPosition	EnvObject	Loop	High 0,82	Very High	Very High
5	AttachTo	Deposit	DepositPosition	EnvObject	Stop	High 0,82	Very High	Very High
7	*Err 2*	*Move*	*MoveLocation*	*EnvObject*	*IfElse*	*Low 0*	*Very High*	*Low*
		If (Position>2) LOOP else STOP						
8	*Err3*	*Detect Position*	*SearchedObject*	*EnvObject*	*ANY*	*Low 0*	*Low*	*Low*
		Move	*MoveLocation*	*EnvObject*				

Table 7.20: Evaluation results for the fulfilment of the process requirement based on the List of Scenarios and an extract of the List of Impossible Scenarios.

requirement fulfilment.

Besides the evaluation of scalability, this qualitative effect analysis has an impact on the validation of Research Task 7 requiring openness for a broad range of functional changes. The results presented in table 7.20 indicate that MobComm implementation is an evaluation platform that does not provide handling mechanisms for impossible or inconsistent New Skill Descriptions. Future work needs to enhance the provided implementation to a level of robustness where all scenarios presented above can be classified with a *Very High* process requirement fulfilment.

For the List of Scenarios, however, the process fulfilment can be completely assessed with *Very High*. As the compliance of this list is in the main focus of the evaluation, the corresponding results validate a *Very High* process requirement fulfilment in MobComm. In combination with the evaluated lack of scalability, this results in a partial compliance with Research Task 7.

Flexibility

The evaluation of flexibility contains the input variables self-organisation and process

requirement fulfilment as introduced in the evaluation catalogue. The measurement is split into two experiments that provide information about the flexibility to reconfigure new functionalities within the given hardware limits and after hardware changes.

In the first part, self-organisation and process requirement fulfilment are measured for the List of Scenarios with a subsequent application of fuzzy rules (cf. table 7.7). These results as presented in table 7.21(a) map the process-related flexibility of the system that can be set to an average value of *Very High* for all scenarios (with the range from *High* to *Very High*). This value is completed by two experiments that make a statement about the reconfiguration flexibility in case of changed hardware in the forefront of a process change.

(a) Results of the process-related flexibility evaluation.

Skill Name	New Skill Description (NSD)				t_a t_{recon}	Self-organisation	Process fulfilment	Flexibility
	UsedSkill	Precondition	Value	Event				
Follow 1	Detect Location	SearchedObject	EnvObject	Loop	11,24 sec 62 sec	Medium	Very High	High
	Move	MoveLocation	EnvObject					
Follow 2	Move	MoveLocation	EnvObject	Loop	13,90 sec 60 sec	High	Very High	Very High
Follow Until	Move	MoveLocation	EnvObject	IfElse	14,9 sec 55,5 sec	Medium	Very High	High
	If (Location>2) LOOP else STOP							
Tracked Gripping	Pick	PickPosition	EnvObject	Loop	15,0 sec 36 sec	Very High	Very High	Very High
AttachTo	Deposit	DepositPosition	EnvObject	Stop	14,6 sec 37 sec	High	Very High	Very High

(b) Results of the hardware-related flexibility evaluation.

Name	Agent Action	Pre-condition	Post-condition	$t_a = t_{recon}$	Self-organisation	Process fulfilment	Flexibility
$SA_{barcode}$	Read Barcode	Location	EnvObject	95 min	Low	High	Medium

(c) Results of the flexibility after hardware changes.

Name	New Skill Description (NSD)				t_a t_{recon}	Self-organisation	Process fulfilment	Flexibility
	UsedSkill	Precondition	Value	Event				
C-$SA_{identify}$	Barcode	ReadBarcode	EnvObject	Stop	14,0 sec 31,5 sec	High	Very High	Very High
C-SA_{check}	Detect Location	Searched Object	Location	Stop	13,9 sec 33,0 sec	High	Very High	Very High

Table 7.21: Results of the flexibility evaluation including a process-related and hardware-related aspect.

For this hardware-related flexibility, the List of Changed Hardware is executed as specified in table 7.6(a) with a subsequent evaluation of self-organisation and process requirement fulfilment. According to this list, $SA_{barcode}$ is inserted as a new Atomic Skill Agent to process a barcode label at a *Location* with the goal to send the corresponding *EnvObject*, as presented in table 7.21(b).

In this experiment, the adaptation to the new Atomic Skill Agent $SA_{barcode}$ is executed by an expert user in 95 min ($t_{recon} = t_a = 95\,min$). This period of time includes the following activities:

- Programming of the Skill Agent $SA_{barcode}$ including the required behaviours (40 min).
- Extension of the Resource Agent RA_{sensor} with the functionality to analyse a barcode label at a *Location* (15 min).
- Extension of the socket implementation with a command for the platform camera to analyses barcode labels (15 min).
- Extension of the GUI and the ontology with the new concepts, predicates, and agent actions (10 min).
- Software testing and execution in the real-world/simulation (15 min).

Based on the experiment details, as given in the appendix in table B.12, the fuzzification of the adaptation time results in a *Low*-value for the new Atomic Skill Agent $SA_{barcode}$, while the complexity of user input is classed as *Very High* due to the high programming and testing effort. The corresponding *Low* level of self-organisation and a *High* process requirement fulfilment finally lead to a *Medium* degree of flexibility for the insertion of the new Atomic Skill Agent $SA_{barcode}$ after hardware changes.

The second aspect of the hardware-related flexibility evaluates the process changes that are executed subsequent to the integration of new Atomic Skill Agent $SA_{barcode}$. For this aspect, the scenarios *Identify* and *Check* are regarded, as both use the Atomic Skill Agent $SA_{barcode}$ for the resulting Composite Skill Agent. The criteria of a successful reconfiguration for the Composite Skill Agents $C{-}SA_{identify}$ and $C{-}SA_{check}$ are demonstrated in detail in the appendix in table B.11, while the evaluation of their flexibility is presented in table 7.21(c).

Following the calculation basis in the appendix in table B.12, both Composite Skill Agents result in a *Very High* process requirement fulfilment and a *High* level of self-

organisation. In contrast to the reconfiguration of the Atomic Skill Agent $SA_{barcode}$, the subsequent utilisation of the new agent demonstrates a *Very High* level of flexibility.

Due to the service-based Multi-Agent-System in Standard Holon, new Atomic Skill Agents can be integrated into the system with a *Medium* flexibility that has to be executed once for every new Atomic Skill Agent. This evaluation result for hardware-based reconfiguration gives an outlook on future work where MobComm is extended with a self-organised Resource Layer to enhance this value of hardware-based flexibility. Research Task 4 and Research Task 5 aiming at a configuration-independent and hardware-abstracted system, however, can be completely fulfilled with a *Very High* flexibility for both an initial as well as a changed robot configuration.

Reconfigurability

As introduced in the evaluation catalogue, flexibility is a subset of the measurement of reconfigurability which is enhanced with the effort to execute the reconfiguration. Compared to the input variable *complexity of user input* that is used for the evaluation of self-organisation, the reconfiguration effort is based on the amount of inserted user entries to initiate the reconfiguration of an Atomic or Composite Skill Agent. The amount of programming and testing efforts specifies the assignment to a fuzzy value, as detailed in table 7.22.

Description	Reconfiguration effort
No additional user entries required.	Low
Small amount of entries in natural language.	Medium
Small programming and testing effort.	High
High programming and testing effort.	Very High

Table 7.22: Assignment of the reconfiguration effort to the fuzzy values of M_{quali}.

Besides the reconfiguration effort, the level of flexibility is integrated into the qualitative effect analysis. The reconfigurability for the List of Scenarios can be rated with *High* as a *Medium* reconfiguration effort is required and the flexibility varies only between *High* and *Very High*. Table 7.23(a) shows the individual scenarios and their resulting reconfigurability.

The reconfiguration of the Atomic Skill Agent $SA_{barcode}$ requires a *Very High* effort due to the high level of programming and testing. Combined with the *Medium* level of

flexibility, a *Low* level of reconfigurability results for the hardware-related reconfigurations. In contrast to this weak assessment, those reconfigurations that are executed with the changed robot configuration, provide a *High* degree of reconfigurability due to their *Medium* effort to be integrated into the running system.

(a) Reconfigurability results of the List of Scenarios.

Skill Name	New Skill Description (NSD)				Flexibility	Reconfiguration effort	Reconfigurabilty
	UsedSkill	Precondition	Value	Event			
Follow 1	Detect Location	SearchedObject	EnvObject	Loop	High	Medium	High
	Move	MoveLocation	EnvObject				
Follow 2	Move	MoveLocation	EnvObject	Loop	Very High	Medium	High
Follow Until	Move	MoveLocation	EnvObject	IfElse	High	Medium	High
	If (Location>2) LOOP else STOP						
Tracked Gripping	Pick	PickPosition	EnvObject	Loop	Very High	Medium	High
AttachTo	Deposit	DepositPosition	EnvObject	Stop	Very High	Medium	High

(b) Reconfigurability results regarding the insertion of the Atomic Skill Agent $SA_{barcode}$.

Name	Agent Action	Pre-condition	Post-condition	Flexibility	Reconfiguration effort	Reconfigurabilty
$SA_{barcode}$	Read Barcode	Location	EnvObject	Medium	Very High	Low

(c) Reconfigurability after a changed robot configuration.

Name	New Skill Description (NSD)				Flexibility	Reconfiguration effort	Reconfigurabilty
	UsedSkill	Precondition	Value	Event			
$C\text{-}SA_{identify}$	Barcode	ReadBarcode	EnvObject	Stop	Very High	Medium	High
$C\text{-}SA_{check}$	Detect Location	Searched Object	Location	Stop	Very High	Medium	High

Table 7.23: Evaluation results for system reconfigurability with regard to the initial configuration, a hardware change, and a renewed configuration.

As the functional reconfigurability after process changes is a main contribution of this thesis (cf. section 1.3.1), this evaluation result is of great importance. Based on the outcome shown in table 7.23, the *High* reconfigurability for the List of Scenarios and for the List of Changed Hardware indicates that the system is self-aware of a broad range of functional changes which validates Research Task 6. The *Low* reconfigurability after hard-

ware changes, however, shows the boundaries of the actual MobComm implementation, while the compliance with this task is beyond the scope of the present thesis. System enhancements for compliance with this task are rather integrated into the future work chapter.

7.5 Conclusion

The preceding chapter presented the evaluation of the proposed reconfiguration mechanism that is based on the desired compliance with eight research tasks as introduced in table 2.8 on page 74. The corresponding metrics and their way of measurement was inferred from every research task and resulted in an evaluation catalogue in section 7.2.

Based on the resulting set of experiments, benchmarks, and qualitative effect analysis, the evaluation results were presented in section 7.4. The individual results and the desired values are summarised in table 7.24 including the allocation of metric and research task. As the evaluation metrics are inferred from the research tasks, the results of these metrics are, in turn, able to validate the corresponding task after its evaluation.

While the *High* level of self-organisation complies with the desired requirement in Research Task 1, the evaluation of the loss of productivity results even in a demonstrated increase of productivity of 22,3% through the application of MobComm. A dependable system (Research Task 3), however, cannot be completely complied by the suggested mechanisms. To increase the resulting predictability ($pr = 0,74$), future activities that affect the reconfiguration mechanism and the Validity Check are proposed in the future work chapter.

Research Task 4 and 5 are allocated to the system stability and flexibility, and they are validated by the conducted evaluation with a system stability of $0,84$ and a *Very High* level of flexibility. The *High* reconfigurability that is allocated to Research Task 6 enables the self-awareness of the reconfiguration mechanism and its environment, in contrast to the openness for a broad range of functional changes that cannot be completely complied by the presented work. Even if the process requirement fulfilment is evaluated with *Very High*, the missing compliance of scalability in Reconfiguration Holon disallows an unlimited openness of MobComm for future extensions. Within a common number of robot functionalities, i.e. up to 200 Skill Agents in Standard Holon, the scalability of the system can be demonstrated independent of the chosen level of composition. Finally, the

Metric	Result	Desired Value	Task Description		Compliance
Self-organisation (so)	High	High	Provide a reconfiguration mechanism that realises self-organisation.	Task 1	Yes
Loss of productivity (lp)	- 0,22	0	Provide a reconfiguration mechanism that does not affect the level of productivity during reconfiguration.	Task 2	Yes
Predictability of results (pr)	0,76	1	Provide mechanisms that ensure dependability in the use of new functionalities.	Task 3	Partial
System stability (st),	0,84	0,8	Provide a reconfiguration mechanism that allows hardware abstraction.	Task 5	Yes
Flexibility (fl)	Very High	High	Provide a reconfiguration mechanism that is robot configuration independent.		
Reconfigurability (rf)	High	High	Provide a reconfiguration mechanism that is aware of the limitations of its reconfiguration capabilities.	Task 6	Yes
Scalability (sc)	<<1	<1	Provide a reconfiguration mechanism that is open for a broad range of functional process changes.	Task 7	Partial
Process requirement fulfilment (prf)	Very High	Very High			
Adaptability ($t_{a/recon}$)	t_a= 13,9 sec t_{recon}= 50 sec	t_a < 60 sec t_{vc} < 900 sec	Provide a satisfactory fast adaptability to new processes.	Task 8	Yes

Table 7.24: Summary of the MobComm evaluation results including the desired value and its compliance with the assigned research task.

fast adaptability as desired in Research Task 8 can be overachieved by an adaptation time of 13,9 sec ($t_a = 13,9\ sec$) and a reconfiguration time of 50 sec ($t_{recon} = 50\ sec$) that is far below the desired value in the evaluation catalogue.

With the partial compliance of Research Task 3 respectively Research Task 7 and the total fulfilment of the remaining research tasks, the proposed MobComm reconfiguration mechanism can be approved by the conducted evaluation and follows the thesis contribution of section 1.3.1.

While the marginal productivity loss of MobComm (i.e. 6 parts) during process changes compared to a traditional mobile robot control (i.e. 73 parts) constitutes the most positive result of system evaluation, the unexpected low predictability of reconfiguration enhances the motivation to further investigate a verification mechanism as proposed in the following chapter.

Chapter 8

Conclusion and Future Work

As proposed in chapter 3 to chapter 5, an implementation for the MobComm reconfiguration was introduced in chapter 6. In the system evaluation in the previous chapter a set of quantitative and qualitative evaluation results gave indications about the compliance with the eight research tasks and thus the fulfilment of the contribution that was introduced in section 1.3.1. Both the design chapters and the implementation and evaluation chapters generated a set of enhancements for future optimisations of MobComm. An extract of these activities is presented in section 8.2 the conclusion is drawn in the following section.

8.1 Conclusion

Due to the high mass customisation in car manufacturing, the automotive industry has been facing an intense demand of production flexibility for the last decade. This manufacturing flexibility is investigated in different characteristics as outlined in the literature review in chapter 2. The reaction to hardware failures for example is handled in RIA [Guedemann et al., 2006] while the response to a changing production flow is investigated in approaches such as ADACOR [Leitão and Restivo, 2008].

The reaction to functional process changes, however, is the contribution of the present thesis. By providing both hardware-abstraction and maintenance of productivity during the reconfiguration process, a self-organised and dependable integration of the resulting functionality into the running system is proposed in this work.

To comply with the set goals, a holonic multi-agent system is introduced that is di-

vided into a behaviour-based Standard Holon and cognitive Reconfiguration Holons. Standard manufacturing processes are executed efficiently in Standard Holon due to a service-based communication mechanism, whereas the Reconfiguration Holon consists of different reconfiguration agents that execute a self-organised skill composition by using BDI mechanisms. Within the distributed skill composition, a user-inserted New Skill Description is converted into a new Composite Skill Agent containing the desired robot functionality. To enhance dependability of the new skill, the integration of the Composite Skill Agent into Standard Holon precedes the validation of agent behaviour by a sniffing mechanism.

Besides the mobile commissioning robot that is provided at Audi, the experimental setup contains a simulation environment with a reduced set of functionalities. For the setup of the mobile commissioning robot, a list of assessed industrial requirements has been considered to apply mobile robots in car manufacturing. Both the simulation and the real-world environment served as a platform for system evaluation and demonstrated an extensive compliance with the given research tasks.

Due to the compliance of a specific set of industrial requirements as introduced in the motivation section 1.1, the proposed reconfiguration mechanism can highly contribute to the reconfigurability of industrial mobile robots. These requirements in turn limit the generality of the MobComm approach.

To ignore hardware failures and to totally separate MobComm from control level are key requirements for the design of MobComm as an encapsulated and hardware-abstracted mechanism that allows functional reconfigurability.

The resulting hardware-abstraction, adapted from Holonic Manufacturing Systems, provides the possibility to dynamically change system components by skilled workers without the need of extensive programming. In contrast to the compliance with this important industrial requirement, hardware abstraction limits the reconfiguration possibilities of the system as low-level behaviours cannot be integrated in reconfigured skills. The solution of this contradiction is directed to future work.

The reconfiguration potential of Composite Skill Agents is further limited by the assumption of independent MobComm Skills. The compliance of this requirement allows to map Skill Agents dynamically to the Reconfiguration Holon without the need of a behaviour synchronisation. As the real-world behaviour is therefore limited in the applicable robot skills, the mitigation of this assumption is an open issue for further investigations as well.

To complete the presented approach with the desired generality, MobComm has to be integrated in a manufacturing architecture that provides additional mechanisms for the reaction to events like hardware failures, production flow changes or task execution failures. As presented in chapter 2, relevant architectures for a future integration of MobComm are ADACOR [Leitão and Restivo, 2008] or RIA [Guedemann et al., 2006]. The combination of these approaches with MobComm has the potential to contribute to a surpassing level of reconfigurability while maintaining the requirements set for industrial mobile robots.

In case a robust and dependable software architecture will be available for industrial mobile robots in future, Machine Learning must be adapted to industrial needs. Mobile robots research already provides highly developed approaches in this area such as overviewed in [Smart and Kaelbling, 2002, Stone, 2007].

The provision of a dependable architecture is the prerequisite that individual computational steps in MobComm can be automated by learning strategies. An autonomous update of safety constraints for skill combination by Machine Learning is proposed in future work. The use of Machine Learning was excluded in this thesis as the resulting autonomy and the missing predictability of system behaviour were contrary to the requirements of robustness and maintenance of productivity.

The novel combination of a robust standard execution and a flexible reconfiguration mechanism was evaluated as highly productive in terms of component losses during process change. Thus, system evaluation resulted in an affirmation of the research hypothesis where a missing adaptability to functional process chances was identified. Industrial mobile robots using MobComm are capable to raise productivity during process changes twelvefold compared to sequence-programmed mobile robots. Due to the total separation of reconfiguration tasks, no disadvantages are given during standard process execution.

The evaluation of MobComm motivates the further investigation and optimisation of the presented reconfiguration mechanism for a productive use of industrial mobile robots as suggested in the following.

8.2 Future Work

Based on the evaluation results given in table 7.24 on page 207, Research Task 3 and Research Task 7 cannot be completely complied with the proposed reconfiguration mechanism. According to this deficiency, future work presents extensions of MobComm to

completely fulfil these tasks.

To increase dependability (Task 3) the optimisation of the Validity Check is proposed, followed by the enhancement of the Composite Skill Agent structure and the increase of domain flexibility to broaden the range of functional changes that can be handled by the system (Task 7).

Validity Check optimisation (Task 3)

The optimisation of the Validity Check includes an optimisation of the sniffing mechanism and the integration of safety level validation.

The enhancement of the sniffing mechanism concentrates on an optimised interaction between the *Validator Agent* and the VC-PA. In case the *Process Agent for Validity Check* recognises irregularities during sniffing, these errors are classified and sent to the *Validator Agent* in the Reconfiguration Holon. A classification of sniffing errors can be structured in *initialisation error, undesired skill activation, internal activation error, and resource allocation error*. For the named types of mistakes, the Validator Agent has a programmed handling strategy and reactivates the corresponding parts in the reconfiguration mechanism. If for example an *initialisation error* is reported, the generation of the global pre- and postconditions is repeated by the *Initiator Agent*. The remaining reconfiguration agents must also have plans to react to occurring errors during Validity Check.

Besides the optimised handling of sniffing errors, the safety level validation is desired to be integrated into the Validity Check as described in section 5.1. As has already been suggested in table 5.1 on page 131, safety rules are required in a future Validity Check. The goal of the safety rules is mainly the prohibition of certain configurations and thus the prevention of harmful behaviour. In a first step, system invariants restrict safety-critical combinations of skills or resources. For example, the movement of the robot arm always requires a preceding scanning of the environment for localisation. Thus, the activation of the sensory system in the forefront of the arm movement is mandatory. In case this rule is broken, the error is reported to the Validator Agent.

In addition to the prevention of safety-critical configurations, the mechanical and control policies of the hardware system are mapped to the VC-PA. For the mobile commissioning system, as presented in figure 7.6, these policies are the mechanical limits of the robot arm, the dimensions of the platform as a restriction of movement for the arm, or the dimensions of the gripper as a restriction of working space. During Validity Check the

fulfilment of these policies has to be validated.

Besides the enhancement of dependability through the Validity Check optimisation, the openness for functional changes can be raised with the advancement of the Composite Skill Agent structure as proposed in the following.

Enhancement of Composite Skill Agent structure (Task 7)
The structure of the Composite Skill Agent can be enhanced regarding its openness for a broader range of functional changes. This improvement includes optimised search mechanisms during the Distributed Skill Composition, a refined error handling in Reconfiguration Holon, and the generation of more complex path structures in the resulting Finite State Machine.

In the actual implementation of the skill composition that is executed among the set of *Execution Agents*, a distributed backwards search is applied. For the optimisation of reconfiguration results, the application of alternative or complementary search mechanisms has to be investigated. Further, refinement mechanisms for error handling during reconfiguration must be introduced. According to the handling of sniffing errors, agent plans specify the internal handling of irregularities during reconfiguration. The internal mistakes are classified in *condition deficiency, parameter mismatch, event inconsistency, and interaction abortion*. If, for example, no matching of the required conditions can be found, the *condition deficiency*-plan of the *Initiator Agent* chooses an alternative search mechanism and retries the composition of the new skill.

The generation of more complex path structures for the resulting Finite State Machine can enhance the range of Events in the Composite Skill Agent. As described in section 6.4, the investigation of a more complex Finite State Machine is based on the work of [Goh et al., 2007] where a JADE-FSM-Engine is proposed. Through a dynamic generation of the complete Finite State Machine instead of a state parametrisation, the structure of the Generic Skill Agent is generated in a highly flexible way. As a consequence, the Generic Skill Agent is reduced to a template for a FSM description file which is usable for the JADE-FSM-Engine.

Additionally to the optimisation of search mechanisms and the refined error handling strategies that raise the effectiveness of the actual reconfiguration possibilities, the generation of more complex Finite State Machines allows to broaden the range of inserted process changes as the Events are not limited to the actual setting any more.

The enhanced openness for functional changes is proposed to be combined with an advanced domain flexibility with a dynamic extension of the MobComm ontology in the following.

Enhancement of domain flexibility (Task 7)

As a dynamic generation of Finite State Machines improves the variety of manageable process changes, the enhancement of domain flexibility has to provide the semantic means to insert and process these changes. The domain flexibility is augmented by the dynamic adaptation of the graphical interface to changed processes in addition to the on-line extension of the MobComm ontology.

For the dynamic extension of the MobComm ontology, an ontology generator is investigated as proposed in the WS2JADE approach [Nguyen et al., 2005] in the context of a run-time deployment of Web Services. The ontology generator in MobComm is further initialised by the GUI-agent to integrate new concepts, predicates, or actions into the existing ontology on user demand. In turn, the graphical interface itself is dynamically created by mapping the actual ontology to the user. This allows to dynamically use and create new ontology vocabularies for future reconfiguration processes.

The already presented extensions of MobComm react to a missing compliance in system evaluation, as shown in table 8.1. The subsequent enhancements go beyond the scope of this thesis but constitutes an expedient supplement to the proposed work. To enhance the hardware abstraction and configuration independence (Task 4 and Task 5) of the system, a transfer of the reconfiguration mechanism to the Resource Layer is suggested in the following.

Mechanism transfer to Resource Layer (Task 4 and Task 5)

Based on the evaluation results in section 7.4.2, the hardware-related reconfigurability of the system is *Low* while its compliance is beyond the scope of the thesis. To enhance flexibility and reconfigurability after changed hardware, the transfer of the proposed reconfiguration mechanism to Resource Layer is suggested.

Adapted from the device modelling in the SIARAS approach [Bengel, 2007], a New Resource Description is inserted in the system by the user. By the application of a MobComm API, a standardised socket implementation must still be provided for the integration of any

Task Description		Future Work
Task 3	Provide mechanisms that ensure dependability in the use of new functionalities.	Validity Check optimisation and extension.
Task 4	Provide a reconfiguration mechanism that allows hardware abstraction.	Transfer of the reconfiguration mechanism to Resource Layer.
Task 5	Provide a reconfiguration mechanism that is robot configuration independent.	
Task 7	Provide a reconfiguration mechanism that is open for a broad range of functional process changes.	Optimisation of Composite Agent structure; Enhancement of domain flexibility.

Table 8.1: Proposed system enhancements for future work including their assignment to related research tasks.

new hardware component. The content of this socket implementation is a set of abstracted commands that are used for the self-organised resource allocation in future work.

Within the New Resource Description, the user specifies the sequence of commands and their conditions according to the skill sequence in the New Skill Description. A Reconfiguration Holon further starts a mechanism that allows to combine the provided commands with the desired conditions. Based on a Generic Resource Agent, a Composite Resource Agent is integrated into Standard Holon following the successful reconfiguration and offers a service to Skill Layer as presented in figure 8.1.

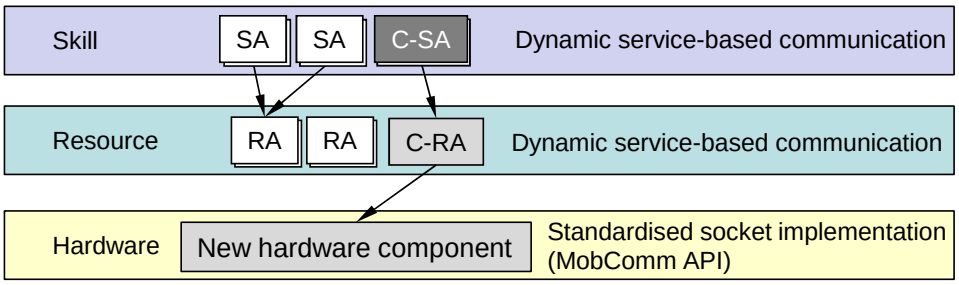


Figure 8.1: Transfer of the reconfiguration mechanism to Resource Layer.

Through application of the reconfiguration mechanism to Resource Layer, hardware-based reconfigurability can be facilitated in MobComm. The reconfiguration effort for the insertion of a new hardware component is limited to the socket implementation based on the MobComm API. After the provision of the socket commands, the Composite Resource Agent that offers a new service to Skill Layer, and the corresponding Composite Skill Agent can be reconfigured self-organised in the system.

Besides the already named enhancements of MobComm, a set of open questions is

given for further investigations of dependable and flexible industrial mobile robots.

An open issue is the mitigation of skill independence without loosing the functional reconfigurability of the robot. Mechanisms have to be investigated that are able to handle skill synchronisation and functional process changes in an industrial environment.

Besides the relaxation of limiting MobComm assumptions, a formal behaviour verification is directed to future work. As alternatives to the proposed Validity Check, the use of Petri Nets (cf. ADACOR [Leitão and Restivo, 2008]) or Linear Temporal Logic (cf. RIA [Guedemann et al., 2006]) has to be investigated. The challenge to be accepted is to sustain the robustness of standard execution and the easy change of hardware components meanwhile new functionalities can be formally verified and dependably integrated.

The detailed MobComm enhancements and the set of open questions contribute to dependable industrial mobile robots that can dynamically react to a wide range of environmental changes.

The adaptation of promising research in the area of mobile robots to real-world applications in industry is an ongoing topic and a hopefully growing interest within the robotics community. The proposed enhancements of MobComm are a further step towards highly flexible and dependable industrial mobile robots in future.

Appendix A

Implementation Details

```
protected ACLMessage handleRequest(ACLMessage request)
{       ...
  //Get the content of the received REQUEST message and handle exceptions.
  contentElement = myAgent.getContentManager().extractContent(request);
  ...
  boolean validParameters = false;
  if(contentElement instanceof PackageData)
  { ...
  //Check if AnyConcept is packed in PackageData
  PackageData pd= (PackageData)contEle;

  if(pd.getApplicationPackage().size()==1)
  {AppPackage ap=(AppPackage)pd.getAppPackage().get(0);

  if(ap.getApp_obj() instanceof AnyConcept)
  {   validParameters    = true;
  getDataStore().put("AnyConcept", pd);
  }}}

  if(!validParameters)
  {
    //Message not valid, create REFUSE message.
    ACLMessage replyRefuse = request.createReply();
    replyRefuse.setPerformative(ACLMessage.REFUSE);
    return replyRefuse;
  }else{
    //Message valid, create AGREE message.
    getDataStore().put("requesterMsg", request);
    getDataStore().put(REQUEST_KEY, request);
    ACLMessage replyAgree = request.createReply();
    replyAgree.setPerformative(ACLMessage.AGREE);
    ...
    return replyAgree;
  }
    return reply;
}
```

Listing A.1: Code extract of the *HandleRequest*-behaviour of an Atomic Skill Agent.

```
1    public void action() {
2      ACLMessage msg = myAgent.receive();
3      if(msg == null){block();
4      }else{
5          /**
6          * This behaviour waits until a New Skill Input Data is received.
7          *(Exceptions are caught.)
8          */
9          ...
10         ContentElement ce = myAgent.getContentManager().extractContent(msg);
11
12         if(ce instanceof NewSkillInputData)
13         {
14           //New Skill Input Data is stored in the agent.
15           NewSkillInputData nsid = (NewSkillInputData)ce;
16           myAgent.setMyIns(nsid);
17           ...
18
19           //Skill Agent knowledge is extracted from the NSID.
20           myAgent.setMyFsm(nsid.getFsmStateDescriptions());
21           myAgent.setMyName(nsid.getSkillName());
22           myAgent.setDataMappings(masHashMapToHashMap(nsid.getMasHashMap()));
23           myAgent.setMyCond(nsid.getConditionElement());
24
25           //Extract the Event related States of the Skill Agent.
26           ...
27           for (Iterator iterator =nsid.getAllFsmStateDescriptions();
28           iterator.hasNext();)
29           {fsd = (FSMStateDescription) iterator.next();
30           if (fsd.getConditionFlag())
31           {myAgent.setMyCondState(fsd.getStateName());
32           condition = fsd.getStateName();
33           }}
34           //Add the FSMBehaviour to the Skill Agent.
35           myAgent.addBehaviour(new GenericSkillBehaviourFSM
36                   (myAgent, nsid.getFsmStateDescriptions()));
37           ...
38           }}}
```

Listing A.2: Code extract of *handleInputNewSkill*-behaviour of a Generic Skill Agent.

```
1  ...
2  import audi.neobotix.datatypes.*;
3  /**
4   *Implementation of the CmdMoveRelativeData command for a
5   *relative movement of the platform.
6   */
7  public class CmdMoveRelativeData extends PltfSocketSendCmdData {
8    private int x, y, angle;
9
10   public CmdMoveRelativeData(int x, int y, int angle) {
11     ...
12     commandId = 22; //See Neobotix Platform API for details.
13   }
14
15   @Override
16   public java.io.ByteArrayOutputStream decode() {
17     return getOutputStreamFromParams(new MobCommDataType[] {
18         new MobCommDataTypeByte(commandId),
19         new MobCommDataTypeInt(x), new MobCommDataTypeInt(y),
20         new MobCommDataTypeInt(angle), });
21   }
22 ...
23 }
```

Listing A.3: Code extract of *CmdMoveRelativeData*-command socket implementation.

```
<?xml version="1.0" encoding="UTF-8"?>
<p:agent <!--Specifications -->>
  <p:imports>
    <p:import>jadex.planlib.*</p:import>
    ...
  </p:imports>

  <p:capabilities>
    <p:capability name="amscap" file="jadex.planlib.AMS" />
  ...
  </p:capabilities>

  <p:beliefs>
    <!-- Store the required ontology concepts or required agents-->
    <p:belief name="ontologyConcept" class="ontologyConcept"/>
    <p:beliefset name="createdAgents" class="AgentIdentifier" />
  ...
  </p:beliefs>

  <p:goals>
    <!-- Goal to perform an action dependent on several conditions-->
    <p:performgoal name="InitialiseSomeOtherAction">
      <p:creationcondition>
        $beliefbase.someNumber != null &amp;&amp;
        ...
      </p:creationcondition>
    </p:performgoal>
  ...
  </p:goals>

  <p:plans>
    <!-Plan that is triggered by a message -->
    <p:plan name="doSomething">
      <p:body class="doSomethingClass"></p:body>
      <p:trigger><p:messageevent ref="receiveContent"/></p:trigger>
    </p:plan>
    <!-Plan that is triggered by a goal-->
    <p:plan name="createSomeAgent">
      <p:body class="ExecutionSomethingPlan"></p:body>
      <p:trigger><p:goal ref="AnalyseData"/></p:trigger>
    </p:plan>
    ...
  </p:plans>

  <p:events>
    <!-- MessageEvent to receive a specific REQUEST message -->
    <p:messageevent name="receiveRequest" type="fipa" direction="receive">
      <p:parameter name="performative" class="String" direction="fixed">
        <p:value>SFipa.REQUEST</p:value>
      </p:parameter>
    ...
    </p:messageevent>
    ...
  </p:events>

  <p:properties>
     <!-- Register the SH-Ontology to the agent to create messages -->
    <p:property name="contentcodec.SHOntology">
      new JadeContentCodec(new SLCodec(),SHOntology.getInstance())
    </p:property>
  ...
  </p:properties>

  <p:configurations>
    <p:configuration name="default">
      <!-- Integrate e.g. initial goals of the agent -->
      ...
    </p:configuration>
  </p:configurations>

</p:agent>
```

Listing A.4: Code extract of a reconfiguration agent skeleton.

Appendix B

Evaluation Details

	Experiment parameters for the evaluation of adaptability				
	Follow 1	Follow 2	FollowUntil	TrackedGrip	AttachTo
Platform Control connected?	Yes	Yes	Yes	Yes	Yes
Arm/Gripper control connected?	Yes	Yes	Yes	Yes	Yes
MobComm is executed on which machine?	Dell Latitude E6400 (32bit Windows)	Dell Latitude E6400 (32bit Windows)	Dell Latitude E6400 (32bit Windows)	Dell Latitude E6400 (32bit Windows)	Dell Latitude E6400 (32bit Windows)
Which processor was used?	Intel Core2 Duo 2.36 GHz	Intel Core2 Duo 2.36 GHz	Intel Core2 Duo 2.36 GHz	Intel Core2 Duo 2.36 GHz	Intel Core2 Duo 2.36 GHz
VC executed in simulation or real-world?	Real-world	Real-world	Real-world	Simulation	Simulation

Table B.1: Evaluation parameters for the experiment to measure the adaptation time t_a and the reconfiguration time t_{recon}.

		No exception is raised during standard execution or reconfiguration. Result?	No deadlock occurs in the system: Reconfiguration is either terminated or successful. Result?
List of Scenarios	Follow 1	No exception.	Successful.
	Follow 2	No exception.	Successful.
	FollowUntil	No exception.	Successful.
	TrackedGrip	No exception.	Successful.
	AttachTo	No exception.	Successful.
List of Impossible Scenarios	Err 1	No exception.	Successful.
	Err 2	No exception.	Successful.
	Err 3	No exception.	Successful.
	Err 4	No exception.	Successful.
	Err 5	No exception.	Successful.
	Err 6	No exception.	Successful.
List of Changed Hardware	Identify	No exception.	Successful.
	Check	No exception.	Successful.

Table B.2: Detailed evaluation of system stability for the List of Scenarios, List of Impossible Scenarios, and List of Changed Hardware.

Description of criteria for a successful reconfiguration	Scenarios of the List of Scenarios				
	Follow 1	Follow 2	FollowUntil	TrackedGrip	AttachTo
New Skill Description is integrated into the Recon-figuration Holon. Yes or No?	Yes	Yes	Yes	Yes	Yes
Used Skills are extracted from the New Skill Description. Result?	SAmove SAdetect	SAmove	SAmove	SAgrip	SAdeposit
Execution Agents are created including the knowledge of the Cloned Skill Agents. Result?	SAmove SAdetect	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit	SAmove, SAgrip, SAdetect, SAdeposit
Condition mismatches are detected and Condition Requests are initialised. Result?	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: EnvObject Required: Location	SAgrip: Desired: EnvObject Required: Position	SAdeposit: Desired: EnvObject Required: Position
Returning Matching Reports are analysed and a New Skill Input Data is created. Resulting FSMState Descripions?	1. State: Cl-SA detect Location 2. State: Cl-SAmove Event: Loop	1.State: Cl-SAdetect Location 2. State: Cl-SAmove Event: Loop	1.State: Cl-SAdetect Location 2. State: Cl-SAmove Event: IfElse at Cl-Sadetect Location	1.State: Cl-SAdetect Position 2.State: Cl-SAgrip Event: Loop	1.State: Cl-SAdetect Position 2.State: Cl-SAdeposit Event: Loop
New Skill Input Data is transformed into a Composite Skill Agent C-SAx. Result?	C-SAfollow	C-SAfollow	C-SA followUntil	C-SA trackedGrip	C-SA attachTo
Validity Check is executed and new Skill is integrated permanently in Standard Holon. Yes or No?	Yes	Yes	Yes	Yes	Yes
Functionality of the Composite Skill Agent matches with the New Skill Description. Result?	Robot follows the transport cart.	Robot follows the transport cart.	Robot follows the transport cart until a distance of 2 meters.	Robot grips a specific EnvObject	Robot attaches an EnvObject in the gripper to another EnvObject

Table B.3: Criteria for a successful reconfiguration while applying the List of Scenarios.

Description of criteria for a successful reconfiguration	Scenarios of the List of Impossible Scenarios					
	Err 1	Err 2	Err 3	Err 4	Err 5	Err 6
New Skill Description is integrated into the Reconfiguration Holon. Yes or No?	Yes	Yes	Yes	No. Reconfiguration terminated.	Yes	Yes
Used Skills are extracted from the New Skill Description. Result?	SAmove	SAmove	SAdetect Position SAmove		SAmove	SAmove Location
Execution Agents are created including the knowledge of the Cloned Skill Agents. Result?	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit.	SAdetect Position Samove		SAmove SAgrip, SAdetect, SAdeposit.	No. Reconfiguration terminated.
Condition mismatches are detected and Condition Requests are initialised. Result?	No. Reconfiguration terminated	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: Position Required: Location		No. Reconfiguration terminated	
Returning Matching Reports are analysed and a New Skill Input Data is created. Resulting FSMState Descripions?		1.State: Cl-SA detect Location 2. State: Cl-SAmove Event: None	No. Reconfiguration terminated!			
New Skill Input Data is transformed into a Composite Skill Agent C-SAx. Result?		C-SA followUntil				
Validity Check is executed and new Skill is integrated permanently in Standard Holon. Yes or No?		Yes				
Functionality of the Composite Skill Agent matches with the New Skill Description. Result?		Robot follows the transport cart until a distance of 2 meters.				

Table B.4: Criteria for a successful reconfiguration while applying the List of Impossible Scenarios.

Killed Agent	Criteria for system stability during the execution of a scenario	
	No exception is raised during standard execution or reconfiguration. Result?	No deadlock occurs in the system: Reconfiguration is either terminated or successful. Result?
GTA	No exception, handled AMS failure.	Terminated.
SAmove	No exception, reconfiguration terminated.	Terminated.
SAdetectLocation	No exception, reconfiguration terminated.	Terminated.
SAgrip, SAdeposit, SAdetectPosition	No influence.	Successful.
I-Eamove	No exception.	Deadlock occurs. Protocol can not be terminated.
I-EAdetectLocation	No exception.	Deadlock occurs. Protocol can not be terminated.
I-SAgrip, I-SAdeposit, I-SAdetectPosition	No influence.	Successful.
I-IA	No exception.	Deadlock occurs. Central point not available any more.
I-VA	No exception, reconfiguration terminated.	Terminated.

Table B.5: Evaluation of system stability after killing specific agents.

Description of criteria for a successful reconfiguration	Killed agents during the use case *Follow transport cart*			
	GTA	SAmove	SAdetect Location	SAgrip, SA deposit, SA detectPosition
New Skill Description is integrated into the Reconfiguration Holon. Yes or No?	Yes	Yes	Yes	Yes
Used Skills are extracted from the New Skill Description. Result?	SAmove	SAmove	SAmove	SAmove
Execution Agents are created including the knowledge of the Cloned Skill Agents. Result?	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, Sadetect,, SAdeposit.	SAmove, SAgrip, Sadetect,, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit.
Condition mismatches are detected and Condition Requests are initialised. Result?	SAmove: Desired: EnvObject Required: Location	No. Reconfiguration terminated!	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: EnvObject Required: Location
Returning Matching Reports are analysed and a New Skill Input Data is created. Resulting FSMState Descripions?	1.State: Cl-SA detectLocation 2. State: Cl-SAmove Event: Loop		No. Reconfiguration terminated!	1.State: Cl-SAdetect Location 2. State: Cl-SAmove Event: Loop
New Skill Input Data is transformed into a Composite Skill Agent C-SAx. Result?	C-SAfollow			C-SAfollow
Validity Check is executed and new Skill is integrated permanently in Standard Holon. Yes or No?	Yes			Yes
Functionality of the Composite Skill Agent matches with the New Skill Description. Result?	Robot follows the transport cart.			Robot follows the transport cart.

Table B.6: Criteria for a successful reconfiguration after Standard Holon agents crash during the use case *Follow transport cart* execution.

Description of criteria for a successful reconfiguration	Killed Agents during use case *Follow transport cart*				
	I-EAmove	I-EA detectLocation	I-SAgrip, I-SAdeposit, I-SA detectPosition	I-IA	I-VA
New Skill Description is integrated into the Reconfiguration Holon. Yes or No?	Yes	Yes	Yes	Yes	Yes
Used Skills are extracted from the New Skill Description. Result?	SAmove	SAmove	SAmove	SAmove	SAmove
Execution Agents are created including the knowledge of the Cloned Skill Agents. Result?	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit.	SAmove, SAgrip, SAdetect, SAdeposit.
Condition mismatches are detected and Condition Requests are initialised. Result?	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: EnvObject Required: Location	SAmove: Desired: EnvObject Required: Location
Returning Matching Reports are analysed and a New Skill Input Data is created. Resulting FSMState Descripions?	No. Deadlock occurs!	No. Deadlock occurs!	1.State: Cl-SA detect Location 2. State: Cl-SAmove Event:	No. Deadlock occurs!	1.State: Cl-SAdetect Location 2. State: Cl-SAmove Event:
New Skill Input Data is transformed into a Composite Skill Agent C-SAx. Result?			C-SAfollow		C-SAfollow
Validity Check is executed and new Skill is integrated permanently in Standard Holon. Yes or No?			Yes		No. Recon-figuration terminated.
Functionality of the Composite Skill Agent matches with the New Skill Description. Result?			Robot follows the transport cart.		

Table B.7: Criteria for a successful reconfiguration after Reconfiguration Holon agents crash during the use case *Follow transport cart* execution.

Single scenarios of the stability experiment		Score / points	Explanation
List of Scenarios	Follow 1	100	Full stability demonstrated.
	Follow 2	100	
	FollowUntil	100	
	TrackedGrip	100	
	AttachTo	100	
List of Impossible Scenarios	Err 1	100	
	Err 2	100	
	Err 3	100	
	Err 4	100	
	Err 5	100	
	Err 6	100	
Agent crashes	GTA	100	
	SAmove	100	
	SAdetectLocation	100	
	SAgrip, SAdeposit, SAdetectPosition	100	
	I-Eamove	20	Artificially caused deadlock!
	I-EAdetectLocation	20	
	I-SAgrip, I-SAdeposit, I-SAdetectPosition	100	Full stability demonstrated.
	I-IA	20	Artificially caused deadlock!
	I-VA	100	Full stability demonstrated.

Table B.8: Calculation details for the total system stability in MobComm.

	Standard process execution “Commissioning cardan shaft”		
	No parallel reconfiguration	7 Execution Agents in RH	200 Execution Agents in RH
Init robot system	6 sec	5 sec	5 sec
Move to Box	9 sec	9 sec	9 sec
Grip a cardan shaft	62 sec	63 sec	61 sec
Move to transport cart	10 sec	8 sec	10 sec
Deposit cardan shaft	65 sec	67 sec	63 sec
Move to start station	16 sec	16 sec	15 sec
Total standard execution time	168 sec	168 sec	163 sec

Table B.9: Execution times for the single steps of the standard process *Commission cardan shafts*.

	Reconfiguration time of scenarios in sec				
Repetitions	Follow 1	Follow 2	FollowUntil	TrackedGrip	AttachTo
1	62	60	55	36	37
2	61	58	58	36	36
3	61	61	58	35	37
4	59	59	57	36	36
5	60	60	55	35	36
6	58	61	56	37	36
7	59	60	56	35	36
8	62	61	59	36	36
9	60	61	59	36	36
10	61	60	58	35	37
Coefficient of variations	0,022	0,013	0,027	0,020	0,013

Table B.10: Calculation basis for the coefficient of variations for the evaluation of the predictability of results.

Description of criteria for a successful reconfiguration	Scenarios of the List of Changed Hardware	
	Identify	Check
New Skill Description is integrated into the Reconfiguration Holon. Yes or No?	Yes	Yes
Used Skills are extracted from the New Skill Description. Result?	SAbarcode	SAdetectLocation
Execution Agents are created including the knowledge of the Cloned Skill Agents. Result?	SAmove SAgrip SAdetect, SAdeposit SAbarcode	SAmove SAgrip SAdetect, SAdeposit SAbarcode
Condition mismatches are detected and Condition Requests are initialised. Result?	SAbarcode: Desired: EnvObject Required: Location	SAdetectLocation: Desired: Location Required: EnvObject
Returning Matching Reports are analysed and a New Skill Input Data is created. Resulting FSMState Descripions?	1. State: CI-SAdetectLocation 2. State: CI-SAbarcode Event: None	1. State: CI-SAbarcode 2. State: CI-SAdetectLocation Event: None
New Skill Input Data is transformed into a Composite Skill Agent C-SAx. Result?	C-SAidentify	C-SAcheck
Validity Check is executed and new Skill is integrated permanently in Standard Holon. Yes or No?	Yes	Yes
Functionality of the Composite Skill Agent matches with the New Skill Description. Result?	Robot is able to identify an EnvObject due to its barcode label.	Robot checks the barcode label at a specific Location.

Table B.11: Criteria for a successful reconfiguration while applying the List of Changed Hardware.

Name	$t_a = t_{recon}$	Fuzzification	Complexity of user input	Self-organisation
$SA_{barcode}$	95 min	Low	Very High	Low

Name	V (t_{recon}(n))	Reconfiguration expectations	Predictability of results
$SA_{barcode}$	0,4	1	0,5

Name	Fuzzification predictability	Matching with NSD	Process requirement fulfilment
$SA_{barcode}$	Medium	Very High	High

Name	t_a t_{recon}	Fuzzification	Complexity of user input	Self-organisation
C-$SA_{identify}$	14,0 sec 31,5 sec	Medium	Low	High

Name	V (t_{recon}(n))	Reconfiguration expectations	Predictability of results
C-$SA_{identify}$	0,013	1	0,83

Name	Fuzzification predictability	Matching with NSD	Process requirement fulfilment
C-$SA_{identify}$	High	Very High	Very High

Name	t_a t_{recon}	Fuzzification	Complexity of user input	Self-organisation
C-SA_{check}	13,9 sec 33,0 sec	Medium	Low	High

Name	V (t_{recon}(n))	Reconfiguration expectations	Predictability of results
C-SA_{check}	0,02	1	0,82

Name	Fuzzification predictability	Matching with NSD	Process requirement fulfilment
C-SA_{check}	High	Very High	Very High

Table B.12: Evaluation details of the flexibility metric.

Bibliography

[Alsafi and Vyatkin, 2010] Alsafi, Y. and Vyatkin, V. (2010). Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. *Robotics and Computer-Integrated Manufacturing*, 26:381–391.

[Alur et al., 2000] Alur, R., Henzinger, T., Lafferriere, G., and Pappas, G. (2000). Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971 –984.

[Angerer and Pooley, 2009] Angerer, S. and Pooley, R. (2009). Dependable reconfiguration of mobile manufacturing systems. In *Proceedings of the 14th International IASTED Conference on Robotics and Applications*.

[Angerer et al., 2010a] Angerer, S., Pooley, R., and Aylett, R. (2010a). MobComm: Using BDI-agents for the reconfiguration of mobile manufacturing systems. In *Proceedings of the 6th IEEE International Conference of Automation Science and Engineering (CASE 2010)*.

[Angerer et al., 2010b] Angerer, S., Pooley, R., and Aylett, R. (2010b). Self-reconfiguration of industrial mobile robots. In *Proceedings of the 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2010)*.

[Angerer et al., 2012] Angerer, S., Strassmair, C., Roettenbacher, M., Staehr, M., and Robertson, N. (2012). Give me a hand - the potential of mobile assistive robots in automotive logistics and assembly applications. In *Proceedings of the 4th Annual IEEE International Conference on Technologies for Practical Robot Applications (TePRA 2012) (to be published)*.

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

[Barata et al., 2005] Barata, J., Camarinha-Matos, L., and Onori, M. (2005). A multi-agent based control approach for evolvable assembly systems. In *Proceedings of the 3rd IEEE International Conference on Industrial Informatics*.

[Barata and Camarinha-Matos, 2003] Barata, J. and Camarinha-Matos, L. M. (2003). Coalitions of manufacturing components for shop floor agility - the CoBASA architecture. *International Journal of Networking and Virtual Organisations*, 2:50–77.

[Barata et al., 2006] Barata, J., Santana, P., and Onori, M. (2006). Evolvable assembly systems: A devolpment roadmap. In *12th IFAC Symposium on Information Control Problems in Manufacturing*.

[Bekey et al., 2008] Bekey, G., Ambrose, R., Kumar, V., Lavery, D., Sanderson, A., Yuh, J., and Zheng, Y. (2008). *Robotics: state of the art and future challenges*. Imperial College Press.

[Bellifemine et al., 2008] Bellifemine, F., Caire, G., Poggi, A., and Rimassa, G. (2008). JADE: A software framework for developing multi-agent applications.Lessons learned. *Information and Software Technology*, 50:10 – 21.

[Bellifemine et al., 2000] Bellifemine, F., Poggi, A., Rimassa, G., and Turci, P. (2000). An Object-Oriented Framework to Realize Agent Systems. In *1st AI*IA/TABOO Joint Workshop "From Objects to Agents": Evolutive Trends of Software Systems*.

[Bellifemine et al., 2007] Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. John Wiley & Sons.

[Bench-Capon, 1998] Bench-Capon, T. (1998). The role of ontologies in the verification and validation of knowledge based systems. In *Proceedings of the 9th International Workshop on Database and Expert Systems Applications*.

[Bengel, 2007] Bengel, M. (2007). Modelling objects for skill-based reconfigurable machines. In *3rd I*PROMS Virtual International Conference*.

[Bengel, 2009] Bengel, M. (2009). Model-based configuration - a workpiece-centred approach. In *ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots*.

[Bernhardt et al., 2008] Bernhardt, R., Surdilovic, D., Katschinski, V., Schreck, G., and Schröer, K. (2008). Next generation of flexible assembly systems. In *Innovation in Manufacturing Networks*. Springer Boston.

[Bi et al., 2008] Bi, Z., Lang, S., Verner, M., and Orban, P. (2008). Development of reconfigurable machines. *The International Journal of Advanced Manufacturing Technology*, 39:1227–1251.

[Bongaerts et al., 2000] Bongaerts, L., Monostori, L., McFarlane, D., and Kádár, B. (2000). Hierarchy in distributed shop floor control. *Computers in Industry*, 43(2):123 – 137.

[Bordini et al., 2006] Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Gomez-Sanz, J., Leite, J., O'Hare, G., Pokahr, A., and Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. In *Informatica 30*, pages 33–44.

[Botía et al., 2004] Botía, J., Hernansáez, J., and Skarmeta, F. (2004). Towards an Approach for Debugging MAS Through the Analysis of ACL Messages. In *Multiagent System Technologies*. Springer Berlin / Heidelberg.

[Bratman, 1987] Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Harvard University Press.

[Braubach et al., 2004] Braubach, L., Pokahr, A., and Lamersdorf, W. (2004). Jadex: A short overview. In *Main Conference Net. ObjectDays*.

[Braubach et al., 2008] Braubach, L., Pokahr, A., and Lamersdorf, W. (2008). A universal criteria catalog for evaluation of heterogeneous agent development artifacts. In Jung, B., Michel, F., Ricci, A., and Petta, P., editors, *From Agent Theory to Agent Implementation*, pages 19–28.

[Brennan and Norrie, 2003] Brennan, R. and Norrie, D. (2003). *From FMS to HMS*, chapter 3, pages 31–52. Springer-Verlag.

[Brooks, 1990] Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15.

[Bussmann and Schild, 2000] Bussmann, S. and Schild, K. (2000). Self-organizing manufacturing control: An industrial application of agent technology. In *Proceedings of the 4th International Conference on Multi-Agent Systems*.

[Cakar et al., 2007] Cakar, E., Mnif, M., Mueller-Schloer, C., Richter, U., and Schmeck, H. (2007). Towards a quantitative notion of self-organisation. In *IEEE Congress on Evolutionary Computation*.

[Cass et al., 2001] Cass, A. G., Ramamritham, K., and Osterweil, L. J. (2001). Exploiting hierarchy for planning and scheduling. Technical report, University of Massachusetts Amherst, MA, USA, Amherst, MA, USA.

[Chevaleyre et al., 2006] Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J., Phelps, S., Rodríguez-Aguilar, J. A., and Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica*, 30:3–32.

[Chip Online, 2010] Chip Online (2010). Hype of tomorrow: 3d printer and mobile robots (http://business.chip.de, translated from german).

[Clement et al., 2007] Clement, B. J., Durfee, E. H., and Barrett, A. C. (2007). Abstract reasoning for planning and coordination. *J. Artif. Int. Res.*, 28(1):453–515.

[Collier, 2002] Collier, R. W. (2002). *Agent Factory: A Framework for the Engineering of Agent-Oriented Applications*. PhD thesis, Department of Computer Science, National University of Ireland.

[Colombo et al., 2005] Colombo, A., Schoop, R., and Neubert, R. (2005). An agent-based intelligent control platform for industrial holonic manufacturing systems. *IEEE Transactions on Industrial Electronics*, 53(1):322 – 337.

[Committee on Visionary Manufacturing Challenges, 1998] Committee on Visionary Manufacturing Challenges (1998). *Visionary Manufacturing Challenges for 2020*. National Academies Press.

[Correia, 2006] Correia, L. (2006). Self-organised systems: fundamental properties. *Revista de Ciencias da Computacao*, 1(1):1–10.

[Dashchenko, 2006] Dashchenko, A. I. (2006). *Reconfigurable manufacturing systems and transformable factories*. Springer, 1 edition.

[De Wolf and Holvoet, 2005] De Wolf, T. and Holvoet, T. (2005). Emergence versus self-organization: Different concepts but promising when combined. In Brueckner, S., editor, *ESOA 2004*. Springer-Verlag Berlin Heidelberg.

[Dilts et al., 1991] Dilts, D. M., Boyd, N. P., and Whorms, H. H. (1991). The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems*, 10(1):79 – 93.

[Dimou et al., 2007] Dimou, C., Symeonidis, A., and Mitkas, P. (2007). Towards a Generic Methodology for Evaluating MAS Performance. In *Proceedings of the Conference on Integration of Knowledge Intensive Multi-Agent Systems*.

[Douglass, 1997] Douglass, B. P. (1997). *Real-Time UML: Developing Efficient Objects for Embedded Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Duffie and Piper, 1987] Duffie, N. and Piper, R. (1987). Non-hierarchical control of a flexible manufacturing cell. *Robotics and Computer-Integrated Manufacturing*, 3(2):175–179.

[Durfee and Lesser, 1989] Durfee, E. and Lesser, V. (1989). Negotiating Task Decomposition and Allocation Using Partial Global Planning. *Distributed Artificial Intelligence*, 2:229–244.

[Edwards et al., 2009] Edwards, G., Garcia, J., Tajalli, H., Popescu, D., Medvidovic, N., Sukhatme, G., and Petrus, B. (2009). Architecture-driven self-adaptation and self-management in robotics systems. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*.

[EUPASS, 2008] EUPASS (2008). EUPASS adaptive assembly roadmap 2015 (deliverable 1.5f). Technical report, Project Report-Public Document 1.5f, NMP-2-CT-2004-507978.

[Evermann and Fang, 2010] Evermann, J. and Fang, J. (2010). Evaluating ontologies: Towards a cognitive measure of quality. *Information Systems*, 35(4):391 – 403.

[Fainekos et al., 2009] Fainekos, G. E., Girard, A., Kress-Gazit, H., and Pappas, G. J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352.

[Feng et al., 2007] Feng, Q., Bratukhin, A., Treytl, A., and Sauter, T. (2007). A flexible multi-agent system architecture for plant automation. In *Proceedings of the 5th IEEE International Conference on Industrial Informatics*.

[Ferber, 1999] Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, volume o. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[FIPA, 2001] FIPA (2001). *Agent Communication Language Specifications, http://www.fipa.org/specs*.

[FIPA, 2005] FIPA (2005). *FIPA Interaction Protocol Specifications, http://www.fipa.org/specs/*.

[FIPA, 2011] FIPA (2011). IEEE Foundation for Intelligent Physical Agents (http://www.fipa.org).

[Firby, 1989] Firby, R. J. (1989). *Adaptive execution in complex dynamic worlds*. PhD thesis, Faculty of Graduate School, New Haven, CT, USA.

[Firby, 1996] Firby, R. J. (1996). Modularity issues in reactive planning. In *Proceedings of the Third International Conference on AI Planning Systems*.

[Fletcher et al., 2001] Fletcher, M., Brennan, R., and Xu, Y. (2001). How intelligent manufacturing holons configure themselves. In *Proceedings of the International Conference on Intelligent Systems and Control*.

[Fox, 1994] Fox, M. (1994). ISIS: A Retrospective. *Interlligent Scheduling*, 3(28).

[Frei, 2010] Frei, R. (2010). *Self-organisation in evolvable assembly systems*. PhD thesis, Faculty of Science and Technology, Universidade Nova de Lisboa, Portugal.

[Frei et al., 2007a] Frei, R., Barata, J., and Onori, M. (2007a). Evolvable production systems context and implications. In *IEEE International Symposium on Industrial Electronics (ISIE 2007)*, pages 3233 –3238.

[Frei et al., 2007b] Frei, R., Barata, J., and Serugendo, G. (2007b). A complexity theory approach to evolvable production systems. In *Proceedings of the 3rd International Workshop on Multi-Agent Robotic Systems in conjunction with ICINCO 2007*.

[Frei et al., 2009] Frei, R., Ferreira, B., Di Marzo Serugendo, G., and Barata, J. (2009). An architecture for self-managing evolvable assembly systems. In *SMC'09: Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics.*

[Frei et al., 2007c] Frei, R., Ribeiro, L., Barata, J., and Semere, D. (2007c). Evolvable assembly systems: Towards user friendly manufacturing. In *Proceedings of the 2007 IEEE International Symposium on Assembly and Manufacturing.*

[Frei et al., 2008] Frei, R., Serugendo, G. D. M., and Barata, J. (2008). Designing self-organization for evolvable assembly systems. In *Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems.*

[Gat, 1992] Gat, E. (1992). Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the 10th national conference on Artificial intelligence.*

[Gat, 1998] Gat, E. (1998). *Three-layer architectures.* MIT Press, Cambridge, MA, USA.

[Georgiadis et al., 2002] Georgiadis, I., Magee, J., and Kramer, J. (2002). Self-organising software architectures for distributed systems. In *Proceedings of the first workshop on Self-healing systems.*

[Giret and Botti, 2004] Giret, A. and Botti, V. (2004). Holons and agents. *Journal of Intelligent Manufacturing*, 15:645–659.

[Goh et al., 2007] Goh, S., Chhetri, M. B., and Kowalczyk, R. (2007). JADE-FSM-Engine: A deployment tool for flexible agent behaviours in JADE. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology.*

[Grasse, 1959] Grasse, P. P. (1959). La reconstruction du nid et les interactions inter-individuelles chez les bellicositermes natalenis et cubitermes sp. la theorie de la stigmergie: essai d'interpretation des termites constructeurs. *Insectes Sociaux*, 6:41–83.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

[Guedemann et al., 2008] Guedemann, M., Nafz, F., Ortmeier, F., Seebach, H., and Reif, W. (2008). A specification and construction paradigm for organic computing systems. In *Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems.*

[Guedemann et al., 2006] Guedemann, M., Ortmeier, F., and Reif, W. (2006). Safety and dependability analysis of self-adaptive systems. In *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation.*

[Gupta and Goyal, 1989] Gupta, Y. P. and Goyal, S. (1989). Flexibility of manufacturing systems: Concepts and measurements. *European Journal of Operational Research*, 43(2):119 – 135.

[Hadeli et al., 2003] Hadeli, K., Valckenaers, P., Zamfirescu, C. B., Brussel, H. V., Germain, B. S., Holvoet, T., and Steegmans, E. (2003). Self-organising in multi-agent coordination and control using stigmergy. In *AAMAS 2003 Workshop on Engineering Self-Organising Systems.*

[HaiHua and MiaoLiang, 2007] HaiHua, L. and MiaoLiang, Z. (2007). MAS4AMR: A self organized multi agent system designed for auto mobile robot. In *Proceedings of the Third International Conference on Natural Computation.*

[Helsinger and Wright, 2005] Helsinger, A. and Wright, T. (2005). Cougaar: A robust configurable multi agent platform. In *2005 IEEE Aerospace Conference.*

[Henkel & Roth, 2008] Henkel & Roth (2008). Mobile robot (http://www.henkel-roth.com/mobile-robot.html).

[Huhns and Buell, 2002] Huhns, M. N. and Buell, D. A. (2002). Trusted autonomy. *IEEE Internet Computing*, 6(3):78–80.

[International Organization for Standardization, 2006] International Organization for Standardization (2006). Iso 13849-1:2006 (http://www.iso.org/).

[Jennings, 1999] Jennings, N. R. (1999). Agent-Oriented Software Engineering. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering.*

[Kanchanasevee et al., 1997] Kanchanasevee, P., Biswas, G., Kawamura, K., and Tamura, S. (1997). Contract-net based scheduling for holonic manufacturing systems. In *Proceedings of SPIE, Architectures, Networks, and Intelligent Systems for Manufacturing Integration.*

[Kephart and Chess, 2003] Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer*, 36(1):41 – 50.

[Kim et al., 2006] Kim, D., Park, S., Jin, Y., Chang, H., Park, Y.-S., Ko, I.-Y., Lee, K., Lee, J., Park, Y.-C., and Lee, S. (2006). SHAGE: a framework for self-managed robot software. In *Proceedings of the 2006 international workshop on Self-adaptation and Self-managing systems*.

[Kim and Robertazzi, 2006] Kim, S.-H. and Robertazzi, T. (2006). Modeling mobile agent behavior. *Computers & Mathematics with Applications*, 51(6-7):951 – 966.

[Klostermeyer and Klemm, 2003] Klostermeyer, A. and Klemm, E. (2003). PABADIS - an agent based flexible manufacturing concept. In *Proceedings of the IEEE International Conference on Industrial Informatics*.

[Kollingbaum et al., 2000] Kollingbaum, M., Heikkila, T., Peeters, P., Matson, J., Valckenaers, P., Mcfarlane, D., and Bluemink, G. (2000). Emergent Flow Shop Control Based On Mascada Agents. In *Proceedings of IFAC Symposium on manufacturing, modeling, management and control.*, Patras, Greece.

[Konolige and Myers, 1996] Konolige, K. and Myers, K. L. (1996). The Saphira Architecture for Autonomous Mobile Robots. In Kortenkamp, D., Bonasso, R. P., and Murphy, R., editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press.

[Koren et al., 1999] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., and Brussel, H. V. (1999). Reconfigurable manufacturing systems. *CIRP Annals - Manufacturing Technology*, 48(2):527 – 540.

[Kulic and Nakamura, 2010] Kulic, D. and Nakamura, Y. (2010). Incremental learning of human behaviors using hierarchical hidden markov models. In *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pages 4649–4655.

[Kumar and Cohen, 2004] Kumar, S. and Cohen, P. R. (2004). STAPLE: An Agent Programming Language Based on the Joint Intention Theory. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*.

[Leitao, 2009] Leitao, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7):979 – 991.

[Leitao, 2011] Leitao, P. (2011). A holonic disturbance management architecture for flexible manufacturing systems. *International Journal of Production Research*, 49(5):1269–1284.

[Leitão et al., 2006] Leitão, P., Colombo, A. W., and Restivo, F. (2006). A formal specification approach for holonic control systems: the ADACOR case. *International Journal of Manufacturing Technology and Management*, 8:37–57.

[Leitão and Restivo, 2008] Leitão, P. and Restivo, F. (2008). Implementation of a holonic control system in a flexible manufacturing system. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(5):699–709.

[Leitão, 2004] Leitão, P. (2004). *An Agile and Adaptive Holonic Architecture for Manufacturing Control*. PhD thesis, Department of Electrotechnical Engineering, University of Porto, Portugal.

[Lepuschitz et al., 2010] Lepuschitz, W., Zoitl, A., Vallee, M., and Merdan, M. (2010). Toward self-reconfiguration of manufacturing systems using automation agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews.*, 41:1 –18.

[Lockemann and Nimis, 2009] Lockemann, P. and Nimis, J. (2009). Dependable multiagent systems: Layered reference architecture and representative mechanisms. In *Safety and Security in Multiagent Systems*. Springer Berlin / Heidelberg.

[Lopatkin, 2008] Lopatkin, I. (2008). Resilience through dynamic reconfiguration in agent systems. In *Proceedings of the 2008 RISE/EFTS Joint International Workshop on Software Engineering for Resilient Systems*.

[Louis and Martinez, 2006] Louis, V. and Martinez, T. (2006). Agent communication II. In *An Operational Model for the FIPA-ACL Semantics*. Springer-Verlag.

[Luck et al., 2005] Luck, M., McBurney, P., Shehory, O., and Willmott, S. (2005). *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink.

[Lueder et al., 2005] Lueder, A., Klostermeyer, A., Peschke, J., Bratoukhine, A., and Sauter, T. (2005). Distributed automation: PABADIS versus HMS. *IEEE Transactions on Industrial Informatics*, 1:31 – 38.

[Malec et al., 2007] Malec, J., Nilsson, A., Nilsson, K., and Nowaczyk, S. (2007). Knowledge-based reconfiguration of automation systems. In *IEEE International Conference on Automation Science and Engineering*.

[Mani et al., 2008] Mani, N., Garousi, V., and Far, B. (2008). Monitoring Multi-Agent Systems for deadlock detection based on UML models. In *Canadian Conference on Electrical and Computer Engineering*.

[Mehrabi et al., 2002] Mehrabi, M. G., Ulsoy, A. G., Koren, Y., and Heytler, P. (2002). Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing*, 13(2):135–146.

[Mosemann and Wahl, 2001] Mosemann, H. and Wahl, F. (2001). Automatic decomposition of planned assembly sequences into skill primitives. *IEEE Transactions on Robotics and Automation*, 17(5):709 –718.

[Motta and Lu, 2000] Motta, E. and Lu, W. (2000). A Library of Components for Classification Problem Solving. In *Proeedings of the 6th Pacific International Knowledge Acquisition Workshop*.

[Nafz et al., 2009] Nafz, F., Ortmeier, F., Seebach, H., Steghofer, J.-P., and Reif, W. (2009). A generic software framework for role-based organic computing systems. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 96 –105.

[Nau et al., 2004] Nau, D., Ghallab, M., and Traverso, P. (2004). *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Naumann et al., 2007] Naumann, M., Wegener, K., and Schraft, R. D. (2007). Control Architecture for Robot Cells to Enable Plug'n'Produce. In *IEEE International Conference on Robotics and Automation*, pages 287–292.

[Nguyen et al., 2005] Nguyen, X., Kowalczyk, R., Chhetri, M., and Grant, A. (2005). WS2JADE: a tool for run-time deployment and control of web services as JADE agent services. In Unland, R., Calisti, M., Klusch, M., Walliser, M., Brantschen, S., Calisti, M., and Hempfling, T., editors, *Software Agent-Based Applications, Platforms and Development Kits*. Birkhaeuser Verlag.

[Nowostawski et al., 2000] Nowostawski, M., Bush, G., Purvis, M., and Cranefield, S. (2000). Platforms for agent-oriented software engineering. In *Proceedings of the Seventh Asia-Pacific Software Engineering Conference 2000.*

[Onori et al., 2006] Onori, M., Barata, J., and Frei, R. (2006). Evolvable assembly systems basic principles. *Information Technology for Balanced Manufacturing Systems*, 220:317–328.

[Paetz, 2002] Paetz, J. (2002). A note on core regions of membership functions. In *European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems.*

[Paprzycki et al., 2004] Paprzycki, M., Abraham, A., Prvanescu, A., and Badica, C. (2004). Implementing agents capable of dynamic negotiations. In *Proceedings of Symbolic and Numeric Algorithms for Scientific Computing.*

[Parker, 2008] Parker, L. (2008). Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems. *Journal of Physical Agents*, 2(1):5–14.

[Peschke et al., 2005] Peschke, J., Lueder, A., and Kuhnle, H. (2005). The PABADIS'PROMISE architecture - a new approach for flexible manufacturing systems. In *Proceedings of the 10th IEEE Conference onEmerging Technologies and Factory Automation, 2005.*

[Plaku, 2008] Plaku, E. (2008). *From high-level tasks to low-level motions: motion planning for high-dimensional nonlinear hybrid robotic systems.* PhD thesis, RICE UNIVERSITY, Houston, TX, USA.

[Pokahr et al., 2003] Pokahr, A., Braubach, L., and Lamersdorf, W. (2003). Jadex: Implementing a BDI-infrastructure for JADE agents. *EXP - in search of innovation (Special Issue on JADE)*, 3:76–85.

[Pokahr et al., 2005] Pokahr, A., Braubach, L., and Lamersdorf, W. (2005). Jadex: A BDI Reasoning Engine. In R. Bordini, M. Dastani, J. D. and Seghrouchni, A. E. F., editors, *Multi-Agent Programming*, pages 149–174. Springer Science+Business Media Inc., USA. Book chapter.

[Pollard et al., 2008] Pollard, D., Chuo, S., and Lee, B. (2008). Strategies for mass customization. *Journal of Business & Economics Research*, 6(7):77–86.

[Post et al., 1997] Post, W., Wielinga, B., de Hoog, R., and Schreiber, G. (1997). Organizational Modeling in CommonKADS: The Emergency Medical Service. *IEEE Expert: Intelligent Systems and Their Applications*, 12:46–52.

[Profactor GmbH, 2010] Profactor GmbH (2010). Toolkit for building low cost robot coworkers in assembly lines (http://www.locobot.eu/).

[Pěchouček and Mařík, 2008] Pěchouček, M. and Mařík, V. (2008). Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17(3):397–431.

[Rao and Georgeff, 1991] Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning.*

[Richter et al., 2006] Richter, U., Mnif, M., Branke, J., Mueller-Schloer, C., and Schmeck, H. (2006). Towards a generic observer/controller architecture for organic computing. In *INFORMATIK 2006 - Informatik fuer Menschen!*

[Rockwell Automation Inc., 2006] Rockwell Automation Inc. (2006). Java sniffer (http://jade.tilab.com/community-addons.php).

[Rudin, 1997] Rudin, K. (1997). Quantifying scalability (http://www.information-management.com/).

[Schraft et al., 2005] Schraft, R., Meyer, C., Parlitz, C., and Helms, E. (2005). PowerMate - A Safe and Intuitive Robot Assistant for Handling and Assembly Tasks. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.*

[Schreiber et al., 1999] Schreiber, G., Akkermans, H., Anjewierden, A., Dehoog, R., Shadbolt, N., Vandevelde, W., and Wielinga, B. (1999). *Knowledge Engineering and Management: The CommonKADS Methodology.* MIT Press.

[Seebach et al., 2007] Seebach, H., Ortmeier, F., and Reif, W. (2007). Design and construction of organic computing systems. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4215 –4221.

[Semere et al., 2007] Semere, D., Barata, J., and Onori, M. (2007). Evolvable Assembly Systems: Developments and Advances. In *IEEE International Symposium on Assembly and Manufacturing, 2007.*

[Serugendo et al., 2008] Serugendo, G. D. M., Fitzgerald, J., Romanovsky, A., and Guelfi, N. (2008). A Generic Framework for the Engineering of Self-Adaptive and Self-Organising Systems. In Bellman, K., Hinchey, M. G., Müller-Schloer, C., Schmeck, H., and Würtz, R., editors, *Organic Computing - Controlled Self-organization.*

[Serugendo et al., 2003] Serugendo, G. D. M., Foukia, N., Hassas, S., Karageorgos, A., Mostéfaoui, S. K., Rana, O. F., Ulieru, M., Valckenaers, P., and van Aart, C. (2003). Self-organisation: Paradigms and applications. In *Postproceedings of the Engineering Self-Organising Applications workshop at the Second International Joint Conference on Autonomous Agents & Multi-Agent Systems.*

[Serugendo et al., 2006] Serugendo, G. D. M., Gleizes, M.-P., and Karageorgos, A. (2006). Self-Organisation and Emergence in MAS: An Overview. *Informatica*, 30:45–54.

[Shehory et al., 1998] Shehory, O., Sycara, K., Chalasani, P., and Jha, S. (1998). Agent cloning: An approach to agent mobility and resource allocation. *IEEE Communications*, 36(1):58–67.

[Shen and Norrie, 1999] Shen, W. and Norrie, D. (1999). Agent-based systems for intelligent manufacturing: A state-of-the-art survey. *Knowledge and Information Systems*, 1(2):129–156.

[Singh, 1991] Singh, M. P. (1991). Towards a formal theory of communication for multi-agent systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence.*

[Smart and Kaelbling, 2002] Smart, W. D. and Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation.*

[Smith, 1980] Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104– 1113.

[Sousa et al., 2006] Sousa, J., Poladian, V., Garlan, D., Schmerl, B., and Shaw, M. (2006). Task-based adaptation for ubiquitous computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews.*, 36:328 –340.

[Sprunk et al., 2011] Sprunk, C., Lau, B., Paff, P., and Burgard, W. (2011). Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In *Proceedings of the IEEE International Conference on Robotics and Automation, 2011.*

[Spyns et al., 2002] Spyns, P., Meersman, R., and Jarrar, M. (2002). Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17.

[Staab et al., 2004] Staab, H., abd J. Mayer, M. H., Ritter, A., and Schraft, R. (2004). Realisation of agent-based commissioning using Jini technology. In *Proceedings of 2nd International Conference on Industrial Informatics.*

[Stanford Center for Biomedical Informatics Research, 2009] Stanford Center for Biomedical Informatics Research (2009). Protégé ontology editor (http://www.protege.stanford.edu).

[Sterritt, 2005] Sterritt, R. (2005). Autonomic computing. *Innovations in Systems and Software Engineering*, 1:79–88.

[Stone, 2007] Stone, P. (2007). Multiagent learning is not the answer. It is the question. *Artificial Intelligence*, 171:402–05.

[Sykes et al., 2008] Sykes, D., Heaven, W., Magee, J., and Kramer, J. (2008). From goals to components: a combined approach to self-management. In *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems.*

[Tomlin et al., 2003] Tomlin, C., Mitchell, I., Bayen, A. M., and Oishi, M. (2003). Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001.

[Ulieru, 2004] Ulieru, M. (2004). Emerging computing for the industry: Agents, self-organisation and holonic systems. In *Workshop on Industrial Informatics, 2004.*

[Upton, 1992] Upton, D. M. (1992). Flexible structure for computer controlled manufacturing system. *Manufacturing Review*, 5(1):58–74.

[Uschold and Grüninger, 1996] Uschold, M. and Grüninger, M. (1996). Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155.

[Valckenears et al., 2001] Valckenears, P., Brussel, H. V., Kollingbaum, M., and Bochmann, O. (2001). Multi-agent coordination and control using stigmergy applied to manufacturing control. *Multi-agent Systems and Applications, 9th ECCAI Advanced Course and Agent Link's 3rd European Agent Systems Summer School, Selected Tutorial Papers*, 2086:317–334.

[Vallejo et al., 2010] Vallejo, D., Albusac, J., Mateos, J., Glez-Morcillo, C., and Jimenez, L. (2010). A modern approach to multiagent development. *Journal of Systems and Software*, 83(3):467 – 484.

[Van Brussel et al., 1998] Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers In Industry*, 37(3):255–274.

[Wagner, 2002] Wagner, T. (2002). An agent-oriented approach to industrial automation systems. In *Proceedings of the 3rd International Symposium on Multi-Agent Systems, Large Complex Systems and E-Businesses - MALCEB*.

[Wehrli et al., 2008] Wehrli, F., Dufey, S., Chollet, S., and Jacot, J. (2008). A Decision Making Tool for Reconfigurable Assembly Lines - Eupass Project. In *Micro-Assembly Technologies and Applications*. Springer Boston.

[Winikoff et al., 2002] Winikoff, M., Padgham, L., Harland, J., and Thangarajah, J. (2002). Declarative & procedural goals in intelligent agent systems. In *Principles of Knowledge Representation and Reasoning*.

[Wooldridge, 1998] Wooldridge, M. (1998). Agent-based Computing. *Interoperable Communication Networks*, 1(1):71–97.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152.

[Wooldridge et al., 2000] Wooldridge, M., Jennings, N. R., and Kinny, D. (2000). The Gaia Methodology For Agent-Oriented Analysis And Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3:285–312.

[Wright et al., 2001] Wright, W. A., Smith, R. E., Danek, M., and Greenway, P. (2001). A generalisable measure of self-organisation and emergence. In *Proceedings of the International Conference on Artificial Neural Networks*.

[Xuemei, 2007] Xuemei, H. (2007). Implementing manufacturing reconfiguration methodology in multi agent system of reconfigurable assembly line. In *Proceeding of the 2007 IEEE International Conference on Mechatronics and Automation*.

[Zadeh, 1963] Zadeh, L. (1963). On the definition of adaptivity. *Proceedings of the IEEE*, 51:469 – 470.

[Zadeh, 1996] Zadeh, L. (1996). Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103 –111.

[Zambonelli and Omicini, 2004] Zambonelli, F. and Omicini, A. (2004). Challenges and Research Directions in Agent-Oriented Software Engineering. *Autonomous Agents and Multi-Agent Systems*, 9(3):253–283.

[Zhu, 2001] Zhu, H. (2001). Formal specification of agent behaviour through environment scenarios. In *Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers*.