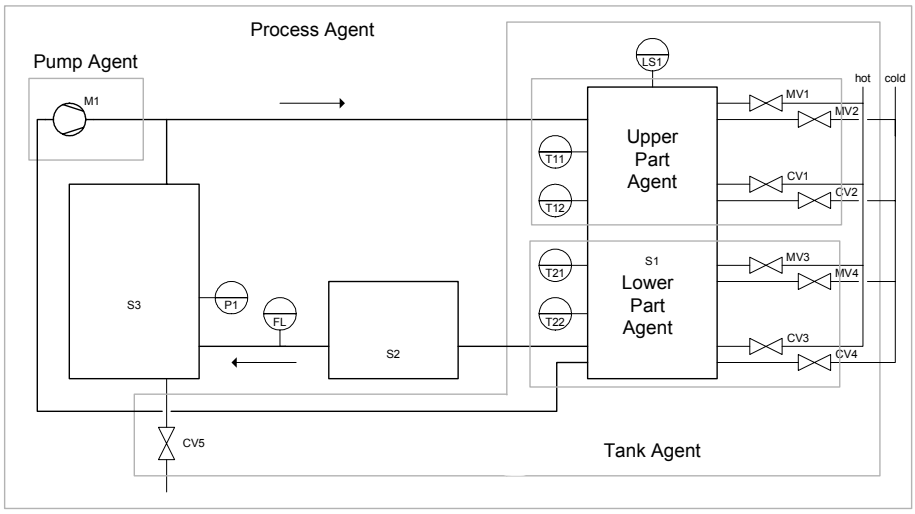
Helsinki University of Technology
Information and Computer Systems in Automation

Espoo 2006 Report 11

An Extended Process Automation System: An Approach based on a Multi-Agent System

Ilkka Seilonen



TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D´HELSINKI

Helsinki University of Technology
Information and Computer Systems in Automation

Espoo 2006 Report 11

An Extended Process Automation System: An Approach based on a Multi-Agent System

Ilkka Seilonen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Automation and Systems Technology, for public examination and debate in Auditorium AS1 at Helsinki University of Technology (Espoo, Finland) on the 10th of February, 2006, at 12 noon.

Helsinki University of Technology
Department of Automation and Systems Technology
Information and Computer Systems in Automation

Distribution:
Helsinki University of Technology
Department of Automation and Systems Technology
Information and Computer Systems in Automation

P.O. Box 5500

FIN-02015 HUT, Finland

Tel. +358-9-451 5462

Fax. +358-9-451 5394

ISBN 951-22-7976-2

ISBN 951-22-7977-0 (PDF)

ISSN 1456-0887

Picaset Oy

Helsinki 2006

HELSINKI UNIVERSITY OF TECHNOLOGY P. O. BOX 1000, FI-02015 TKK http://www.tkk.fi	ABSTRACT OF DOCTORAL DISSERTATION

Author Ilkka Seilonen

Name of the dissertation

An Extended Process Automation System: An Approach based on a Multi-Agent System

Date of manuscript 26.5.2005	Date of the dissertation 10.2.2006
☒ Monograph	☐ Article dissertation (summary + original articles)

Department	Automation and Systems Technology
Laboratory	Information and Computer Systems in Automation
Field of research	Automation technology, information technology
Opponent(s)	Professor Duncan McFarlane, Docent Tapio Heikkilä
Supervisor (Instructor)	Professor Kari Koskinen

Abstract

This thesis describes studies on application of multi-agent systems (acronym: MAS) to enhance process automation systems. A specification of an extended process automation system is presented. According to this specification, MAS can be used to extend the functionality of ordinary process automation systems at higher levels of control. Anticipated benefits of the specification include enhanced reconfigurability, responsiveness and flexibility properties of process automation.

Previous research concerning applications of MAS in process automation has been more limited than in other fields of automation. There has been more research about this topic for example in the area of discrete manufacturing. As goal-oriented distributed systems with coordination capabilities MAS have been found applicable to a part of automation functions, e.g. modification of control logic in abnormal situations. However, when applying MAS to process automation the particular characteristics of this application domain need to be taken into account. The important role of continuous control in process automation needs to be considered.

In this thesis, a specification of an agent platform for process automation is presented as a basis for applying MAS in this application domain. The specification extends a FIPA-compliant agent platform with process automation specific functionality. It utilises a hierarchical agent organisation, a BDI-agent model and qualitative reasoning. It also presents a model for programming MAS applications for process automation with techniques of distributed planning and search. Two applications are specified using the platform. One of these shows how the techniques of distributed planning can be applied in sequential control. The other provides a design model for supervisory continuous control applications using the techniques of distributed search. Experiments performed with a laboratory test environment using prototype implementations of the applications are presented. The experiments are able to demonstrate the feasibility of the approach in limited test scenarios. They also provide information about in which ways MAS techniques are able enhance the properties of process automation.

As a result of the work presented in this thesis more knowledge has been gained about application of MAS in process automation. The specification of the agent platform for process automation and its applications provide a basis for further studies of this topic.

Keywords process automation, multi-agent system, agent platform, BDI-model, distributed planning, distributed search

ISBN (printed)	951-22-7976-2	ISSN (printed)	1456-0887
ISBN (pdf)	951-22-7977-0	ISSN (pdf)	
ISBN (others)		Number of pages	91 p.

Publisher Helsinki University of Technology, Information and Computer Systems in Automation

Print distribution Helsinki University of Technology, Information and Computer Systems in Automation

☒ The dissertation can be read at http://lib.tkk.fi/Diss/2006/isbn9512279770/

Preface

This thesis concerning applications of multi-agent systems in process automation has been prepared at the Laboratory of Information and Computer Systems in Automation at the Helsinki University of Technology (HUT). The work was carried out within two projects, Agent automation (2001-2002) and Mukautuva (2003-2004). These projects were mainly funded by the National Technology Agency (Tekes), which is thankfully acknowledged.

I have become acquainted with the themes of this thesis in the course of several years. I studied planning of control sequences at VTT in the early 90s. Later, I had the opportunity to familiarise myself with agent technology in the CAIP research programme also at VTT. Most of my knowledge about process automation I also gained while working at VTT Automation in 1988-2001. When I joined HUT in 2001 I was able to combine all these themes. However, the experience gained at VTT is clearly reflected in this thesis.

I am very thankful to my supervisor, Professor Kari Koskinen of the Laboratory of Information and Computer Systems in Automation, for first employing me and then supporting me during this work. I also thank Professor Aarne Halme of the Automation Technology Laboratory for supervising the projects and Dr Pekka Appelqvist for his efforts as the project leader. Very big thanks are due to my main co-worker, Mr Teppo Pirttioja. If somebody can find a better co-worker than him then, according to my opinion, he should never quit his job. Many thanks also to Mr Antti Pakonen for his valuable efforts in the Mukautuva project. I also appreciate all my colleagues at the Laboratory of Information and Computer Systems in Automation. Particularly, I have enjoyed the relaxed atmosphere in our coffee room. I also want to thank Mrs Leena Arpiainen for checking the language of this thesis.

However, there are some other things in life than research work. Therefore, I would like to take the opportunity to express my gratitude to my relatives and all of my many friends.

Finally, I would like to say to Wiwin: Terima kasih untuk cinta dan pengertianya.

Espoo 30.11.2005

Ilkka Seilonen

Table of Contents

List of Abbreviations

ACL	Agent communication language
AOSE	Agent-oriented software engineering
API	Application programming interface
BDI	Belief-desire-intention model of software agents
CFP	Call for proposal
CSP	Constraint satisfaction problem
CWS	Chilled water system
DAI	Distributed artificial intelligence
DCS	Distributed control system
DF	Directory facilitator
FIPA	Foundation of Intelligent, Physical Agents
HMS	Holonic Manufacturing Systems
LAN	Local area network
MAS	Multi-agent system
MIMO	Multiple input, multiple output
OPC	OLE for Process Control
PLC	Programmable logic controller
PEM	Plan-execute-monitor model of software agents
WAN	Wide area network
XML	eXtensible markup language

1 Introduction

1.1 Background

The research effort described in this thesis is motivated by three factors. First, there are the requirements of the process automation application domain, i.e. an interest to enhance process automation systems as control systems. Secondly, there is the on-going development of the technologies used for the design and implementation of process automation systems. Thirdly, there is the research of multi-agent systems (acronym: MAS). There is a possibility that MAS might enable enhancement of some properties of process automation systems while utilizing new technical developments as indicated by research in related application domains, e.g. discrete manufacturing. However, there are many open questions concerning the application of MAS to process automation.

The viewpoint to the requirements of process automation systems in this thesis is maybe different from tradition. Usually safety, reliability, quality, efficiency and robustness of control have been regarded as the main requirements in process automation. In this research, the focus is on the reconfigurability, responsiveness and flexibility properties of process automation systems. All these properties are related to adaptation, i.e. management of changes. Reconfigurability means the capability of the automation system to adapt to system configuration changes, i.e. changes in the process equipment and instrumentation. Reconfigurability is an important factor in the maintenance of a production system. The other two properties concern the capability of the automation system to handle different conditions during its operation. Responsiveness refers to reasonable handling of unplanned situations. It includes exception handling and fault tolerance and contributes to the availability of a production system. Flexibility means the capability of the automation system to perform a variety of planned control operations, e.g. producing different products. Process automation systems are rather complex systems which makes it difficult to handle all these types of changes.

Some developments of the technologies used for the implementation of process automation systems are particularly relevant to this thesis. These include increasing distribution of computing power and larger emphasis on software technologies. At the low end of automation intelligent instrumentation and field buses are being adopted to wider usage. At the high end automation systems are increasingly connected to information systems via local area networks. New software technologies e.g. software components have been taken into use in the implementation of automation systems. Web services have recently been studied as a possible systems integration technology. Process automation systems are becoming increasingly distributed, networked and complex software systems.

Multi-agent systems have been a research topic in computer science already for a long time. Although there are several interpretations of the concept of MAS, in this thesis a MAS is considered as one kind of a distributed system consisting of autonomous, goal-oriented and coordinated software modules called agents. MAS:s have been regarded as a possible means for making software systems more adaptable and managing their complexity (Jennings 2001). Agents as autonomous and goal-oriented software modules have been seen as suitable abstractions for decomposing complex

software systems. The autonomy of the agents is to be balanced with appropriate coordination techniques. This arrangement is expected to enable adaptation of operations and eventually lead to a sort of self-organisation in the operation of a MAS. The distribution, complexity management and adaptation properties of MAS seem to match those requirements of process automation that are of interest in this thesis.

Application of MAS to automation systems has previously been studied mainly in the context of discrete part manufacturing. In many cases the motivation of this research has been the possibility to enhance particularly the adaptation and also the complexity management properties of automation. Several results from applications of MAS or similar systems to various control functions of discrete manufacturing have already been published (Marik at al. 2002b, Deen 2003, Bussmann et al. 2004). Some part of the results of this research could be applicable also in process automation. However, because the characteristics of process automation are different from discrete manufacturing there is a need for especially process automation type MAS applications. There has been some research also in this area, but it has been more limited than in discrete manufacturing (Chokshi and McFarlane 2002).

1.2 Research problem

The research problem of this thesis is defined combining the distributed control systems and automation systems development viewpoints.

From the distributed control systems viewpoint the research problem is to develop distributed problem-solving methods for selected process automation control functions assuming that they are implemented with a MAS. This viewpoint emphasises the distributed nature of process automation systems which causes a need for distributed problem-solving. Control operations affecting inter-related control variables but executed by separate controllers need to be coordinated in order to enable meaningful control. The approach to this problem in this thesis is to use the distributed problem-solving methods of MAS. The research task is to specify such MAS-based problem-solving methods that can fulfil the coordination requirements of control operations in process automation and retain the desired properties relating to adaptation. The problem-solving methods need to be selected, applied to the chosen functions and assessed with respect to the requirements.

From the automation systems development viewpoint the research problem is to specify how at least some of the requirements of process automation can be fulfilled by using MAS as a design and implementation methodology. In order to do this, one has to identify some process automation functions first, to which it is both feasible and useful to apply MAS technology. After this, one has to design a MAS that can implement the required functionality. The feasibility and usefulness of this design need to be assessed with respect to the requirements of process automation. The feasibility may be assessed in terms of the capability of the MAS to produce acceptable control actions. The usefulness of the MAS is expected to appear in the adaptation properties of an automation system.

1.3 Research objectives

The objective of this research is to study the research problem, i.e. application of MAS to process automation, from selected viewpoints. The objective is restricted with respect to the studied control operations, applied MAS techniques and assessed system properties. The research objective can be divided into the following three sub-objectives.

- Specification of an agent platform for process automation. This specification defines the role of MAS in process automation with respect to existing automation systems and those aspects of the MAS that are not specific to application types. A design model of a process automation specific MAS is defined in here. This model specifies the MAS techniques to be used, particularly the basic mechanisms of problem-solving in the MAS.

- Specification of control applications based on the platform. Two different control application types are selected for this specification. The application types are sequential control and supervisory continuous control. These are assumed to be suitable to be designed as MAS applications. The specification then defines how the previously specified MAS platform is used in the design of these kinds of applications. In this context utilisation of the process automation specific MAS model and its problem-solving methods to the specific application types is defined.

- Assessment of the platform and the applications. The MAS platform and its applications are to be assessed with respect to their effect on the properties of automation. The properties included in this study are operational correctness as a control system, adaptation-related properties of reconfigurability, responsiveness and flexibility and application development. The objective of the assessment is to identify the mechanisms by which the MAS design affects the properties and recognise the possible limitations of these mechanisms.

1.4 Research methods

This research is conducted as a combination of a literature review, a specification of a MAS for process automation, experimentation with a prototype implementation of the specification in a laboratory test environment and discussion of the MAS properties. The literature review contains surveys of both the research about similar applications of MAS in automation and the selected parts of MAS methodologies to be applied in this research. The specification of the agent platform defines its role with relations to other parts of a process automation system and its own design as a MAS. The prototype implementation is a limited realisation of the specification aimed for laboratory experiments. The purpose of the experiments is to demonstrate the functionality of the MAS applications in restricted test scenarios. The discussion concerns about the relation of this study to related research and the effect of the specification on the properties of automation. Finally, conclusions about the applicability of MAS technology in process automation are drawn based on the discussion.

1.5 Contributions

The contributions of this thesis include the following:

- Specification of a BDI-model-based agent platform for process automation. A model of a process automation-specific MAS is presented. The model defines the role of the MAS as an extension to an ordinary process automation system. It also specifies a hierarchy reflecting the structure of the controlled process as the organisational model of the MAS and the BDI-model as the model of the agents. The basic coordination mechanism in the MAS is the FIPA Contract Net Protocol. The presented specification has many similarities with models published earlier, particularly the ones developed within the Holonic Manufacturing Systems consortium. However, the specification also introduces some design decisions that have not been extensively studied in the area of process automation. The specification emphasises utilisation of peer-to-peer agent relations in coordination. It also uses qualitative process modelling as a knowledge representation mechanism of the agents.

- Specification of a MAS application for sequential control based on the methods of distributed planning. A model of an application for sequential control designed with the previously defined agent platform for process automation is presented. In this model, sequential control is modeled as a distributed planning problem. The specification is based on research in action planning in general and previous studies about agent-based sequential control in particular. Compared to the approaches published earlier, the one in this thesis emphasises decentralisation of planning with utilisation of peer-to-peer relations.

- Specification of a MAS application for supervisory control based on the methods of distributed search. A model of an application for supervisory control designed with the previously defined agent platform for process automation is presented. In this model supervisory control is modeled as a distributed iterative search problem. The specification is based on research in distributed search in general and studies about agent-based continuous control in particular. Compared to the approaches published earlier, the one in this thesis emphasises design of the negotiation agents with the BDI-model also in this application.

- Demonstration of the agent platform and the applications. Prototype versions of the agent platform and both of the applications were implemented. Tests with a laboratory test process were performed in order to demonstrate the operation of the platform and the applications and verifying their functionality in simple test scenarios.

- Discussion of the properties of the agent platform and the applications. The meaning of the presented specifications and experiments are studied by discussions. The properties covered in the discussions include performance of the control operations and the properties related to adaptation of the extended automation system. The presented specifications are also compared to related research. Similarities and differences are identified.

The presented contributions were achieved within a research group where the author has had a central role. The specifications of the agent platform for process automation and the application for distributed planning of control sequences were made almost

exclusively by the author. The specifications of distributed execution of control sequences and distributed search of supervisory control actions were made in co-operation with other members of the research group. The experiments with the test process were done as group work within the research group. Finally, the discussion of the properties of the specifications was performed by the author.

1.6 Outline of the thesis

This thesis contains six chapters organised as follows:

Chapter 1: Introduction.

Chapter 2: Multi-agent systems and process automation. State-of-the-art of the research in MAS is presented as a methodology to be applied in this thesis. Related research in the application of MAS to process automation and related fields is described. Qualitative reasoning is also shortly described as a particular methodology to be applied in this thesis.

Chapter 3: Agent platform for process automation. Specification of the agent platform for process automation is presented both at the agent society and agent levels. An experimental implementation of the platform is described and the properties of the specification are discussed.

Chapter 4: Sequential control based on distributed planning. Studies about developing an application for sequential control with the agent platform are presented. An approach based on distributed planning and plan execution is specified for this purpose. The approach is demonstrated with experiments with the test process. The properties of the approach are discussed.

Chapter 5: Supervisory control based on distributed search. Studies about developing an application for supervisory control with the agent platform are presented. An approach based on iterative distributed search is specified for this purpose. The approach is demonstrated with experiments with the test process. The properties of the approach are discussed.

Chapter 6: Discussion and conclusions. The results obtained in this thesis as a whole are discussed. Based on this, conclusions about the application of MAS to process automation are drawn.

2 Multi-agent systems and process automation

2.1 Introduction

This chapter presents a review of the state-of-the-art in the research of the methodologies and applications to be applied and built upon in subsequent parts of this thesis. The development of the agent platform for process automation (Chapter 3) and the applications for sequential (Chapter 4) and supervisory (Chapter 5) control are based on the methodologies described in this chapter.

The described methodologies and applications include multi-agent systems (acronym: MAS), qualitative process modelling and their applications in process automation. The presentation of MAS methodology focuses on the properties, architecture and selected problem-solving methods of MAS. Problem-solving with distributed search and distributed planning are presented with more details. The application of MAS methodology to automation systems is examined both in the context of process automation and discrete manufacturing. Qualitative reasoning is included in this presentation as one reasoning method of the agents to be used in this research. The end of the chapter contains a discussion about the status of the knowledge concerning MAS applications in process automation.

2.2 Multi-agent systems methodology

2.2.1 Multi-agent systems

In this thesis the concept of an agent is considered from the viewpoint of a MAS. Although the concept of an agent may still lack a generally agreed definition the following one is adopted in this thesis: an agent is an autonomous computational entity capable of flexible and effective operation in order to meet its objectives in its environment (Jennings 2000). When agents cooperate with other agents they form a MAS. MAS can be viewed both as a software engineering paradigm and as a technology. As a paradigm, MAS has its approach for decomposing, abstracting and organizing software systems (Parunak 1997, Jennings 2001). As a technology, MAS provides software engineering methods and tools for design and implementation of systems conforming to its paradigm (Luck at al. 2004). From both viewpoints, the aim of MAS may be regarded to be to facilitate the design and implementation of complex and distributed software systems that are manageable and incorporate flexibility in their behaviour (Jennings 2001). An underlying assumption of MAS is that agents are particularly useful as building blocks for such complex software systems.

The agents in a MAS can be further characterised by studying some of the important properties often required of them.

- Autonomy. Agents have control over their computational resources, state and processes that they encapsulate from other agents (Wooldridge 1999, Jennings 2000). Agents can execute their computational processes independently of other agents, i.e. each agent has its own thread of control.

- Environment. Agents can perceive their environment and react to events in it. They also may affect it by actions (Ferber 1999, Wooldridge 1999, Jennings 2000). The environment can be either physical or computational. With regard to

their environment, the behaviour of agents may be characterised as reactive (Jennings 2000).

- Goal-orientedness. The behaviour of agents is targeted at fulfilling a set of goals (Ferber 1999, Wooldridge 1999, Jennings 2000). The goals for the agents are to be designed by the system designer in such a way that the MAS fulfils its design objectives. Agents can take initiative in order to reach their goals. In a MAS agents can also refuse to fulfil requests of other agents because of their own goals. With regard to their goals, the behaviour of agents may be characterised as proactive.

- Communication. In a MAS the agents can interact with other agents when carrying out their tasks (Ferber 1999, Huhns and Stephens 1999, Jennings 2000). Communication can take place either directly between agents or via some medium. The communication between agents is often expected to take place at knowledge level and relate to the achievement of their goals. With regard to their interaction with other agents, the behaviour of agents may be characterised as social.

- Coordination. In a MAS the agents may coordinate their behaviour with other agents so that a group of agents is able to reach or balance their goals (Huhns and Stephens 1999). Coordination is needed if fulfilling of the goals of different agents is interdependent. Coordination is often assumed to be achieved via communication among the agents and their internal problem-solving.

- Adaptation. Agents are expected to possess some capability to change their behaviour due to various reasons, e.g. their previous performance in fulfilling their goals or changes in their environment or agent society. In a MAS the adaptation may be affected by the interaction between the agents (Sen and Weiss 1999). The agents may learn from each other and change their behaviour as a member of an agent society.

The characterisation of agents and MAS adopted in this thesis has its background in distributed artificial intelligence (acronym: DAI). The agents in DAI are called cognitive agents, whose behaviour is based on symbolic representations of their outside world and their intentions to take some action (Ferber 1999). An opposing approach is based on the purely reactive agents that have minimal or no representations of their outside world and whose behaviour is based on mappings from situations to action (Ferber 1999).

Several coordination mechanisms have been developed for multi-agent systems with different coordination requirements. Coordination among the agents can be based on cooperation or competition (Huhns and Stephens 1999). In the former situation the agents are assumed to be non-antagonistic. In both cases communication has often been considered as a mechanism for enabling coordination. Organisation of agent societies is another generally used means to facilitate coordination in MAS. Negotiation and planning are common among particular coordination mechanisms. Negotiation is particularly suitable for competition situations and planning for cooperation (Huhns and Stephens 1999). Negotiation may also be needed in conjunction with cooperation for conflict resolution. Coordination may also be performed through information sharing, when used to estimate actions of other agents. Significant data structures in coordination include intentions and contracts. Both of

these may be regarded as commitments to act either within one agent or between them.

In a MAS the agents are usually assumed to communicate with each other according to an agent communication language, e.g. FIPA ACL (Labrou 1999, FIPA 2002b). According to this specification, the agents communicate via message passing based on a standardised set of so-called communicative acts with defined formal semantics. Other aspects of agent communication defined in the FIPA standard include content languages and interaction protocols (FIPA 2005). Content languages are used to represent contents of the ACL messages. Interaction protocols define sequences of message exchange forming conversations between the agents.

MAS may be regarded as a developing software engineering paradigm (Jennings and Woolridge 2000, Wooldridge and Ciancarni 1999) that necessitates its own software engineering methods and tools. Research and development work has been conducted both in the area of agent-oriented software engineering (acronym: AOSE) and agent programming tools (Weiss 2002). In the research of AOSE several software engineering methods for agent-based systems have been developed, e.g. Tropos (Castro et al. 2002). Essential results from the development of agent programming tools include several agent platforms, e.g. JADE (JADE 2005) and FIPA-OS (FIPA-OS 2005). The agent platforms provide both application programming interfaces (acronym: API) for application development and run-time environments for executing agent applications. Many of the developed agent platforms are compliant with the FIPA standard.

2.2.2 Agent and multi-agent system architectures

The architecture of a MAS can be studied in the context of one agent and a society of several agents. Agent architecture describes how decision-making about actions is arranged within an agent, i.e. how an agent derives actions from a perceived situation utilizing its internal structure and operation (Wooldridge 1999). Basic types of agent architecture include a reactive agent, a BDI-agent and layered architectures. These models are different e.g. with respect to their complexity, balance between goal-directed and reactive behaviour and response times in various actions.

In a reactive agent decision-making is based on a relatively direct mapping from the perceived situation to action (Wooldridge 1999). A reactive agent may have a representation of a state based on its perceptions. The major advantage of the reactive agent model is its simplicity. The response time of a reactive agent is also easier to estimate than that of more complicated agent models. The main disadvantage of the reactive model is that it may not make the reasoning of the agents explicit.

In a BDI-agent decision-making uses data structures of beliefs, desires and intentions (Wooldridge 1999, Georgeff et al. 1999, Bratman et al. 1988). These concepts represent the knowledge of an agent about its environment, the objectives it tries to achieve and the actions it intends to do in order to achieve the objectives in the perceived situation. During its operation a BDI-agent first updates its beliefs via perception and communication processes and then updates its desires and intentions via option generation and filtering processes. Finally, the agent derives its actions from the current intentions. Although the processing of beliefs, desires and intentions can be based on theorem proving most practical BDI-implementations process the data structures procedurally. Plans containing procedures are a practical method for

representing the logic of the generation and filtering processes. The BDI-architecture enables explicit representation of the reasoning processes of an agent (Singh et al. 1999). However, the BDI-model is more complex than the reactive model. With the BDI-model the response time of an agent is typically more difficult to foresee than with the reactive model.

In a layered agent architecture decision-making of an agent is performed at several layers with different levels of abstraction. The decision-making procedure may also be dissimilar at different layers. Horizontal and vertical layering are two different ways to organise the decision-making layers (Wooldridge 1999). In these models decision-making is organised either in a parallel or sequential fashion. Examples of layered agent architectures are the subsumption architecture with horizontal layering (Brooks 1991) and InteRRaP with vertical layering (Fisher et al. 1994). Layered architectures enable modularisation of different aspects of the decision-making of an agent with possibly dissimilar decision procedures. However, layering may create added complexity and possible inflexibility to the architecture.

In the context of several agents the architecture of a MAS can be characterised using the concept of an agent organisation. It defines different types of agents and their relationships in an agent society. Agent organisations may be described from several viewpoints, e.g. which capabilities the agents have, which part of their environment each agent observes, how information is shared among the agents and how the agents interact between each other (see e.g. Singh and Huhns 2005, Ferber 1999). There are several possible ways how to arrange these aspects each resulting in different types of agent organisations. The agents may have the same capabilities and information or they can be specialised in both senses. In the interaction between the agents different coordination mechanisms may be used. The relations between the agents can form a hierarchical structure with varying numbers of levels. Hierarchical organisations incorporate some level of centralisation as opposed to flat decentralised organisations. Finally, an agent organisation can be either static or emergent with a varying extent of changeable aspects. Depending on their design agent organisations are different with respect to their efficiency, reliability, flexibility and reconfigurability properties.

2.2.3 Distributed search

Search is a common method of problem-solving in artificial intelligence (Russel and Norvig 1995) and mathematical programming (Bazaraa et al. 1993). A search problem may be described with an initial state, a set of operators, a goal test function and a possible path cost function. The task is to find a path from the initial state to the goal state. Path finding problems, constraint satisfaction problems (acronym: CSP) and iterative optimisation are basic types of search problems. Search is one possible problem-solving method for those agents that are expected to do problem-solving. Agents may perform search either locally, i.e. as part of their BDI-model based operation, or as a MAS in the form of a distributed search.

Development of a search application consists of problem formulation and design of a search strategy. In problem formulation the task is to model a problem in the application domain as a search space. States of the search space may correspond to concrete or abstract entities of the application domain. Operators may map e.g. to actions or assignments of values to variables of the application domain. The goal can be a certain state or a function to be optimised. A search strategy is a specification in which order the states of the search space are explored. Domain-specific information

in the form of heuristics is usually useful in the design of a search strategy. Heuristics may estimate distance to the goal state from a given state or just indicate a promising search direction. Also information about the structure of the search space is useful in the design of a search strategy, e.g. breath and depth of a search tree or convexity of the search space. There are several generic search algorithms that can be applied to different types of search problems (see Russel and Norvig 1995, Bazaraa et al. 1993).

Distributed search is a particular case of search in which the search process is performed by several agents. A distributed search problem can be described by stating how the search space is decomposed to different agents. The search space may be shared, separated or partially shared. The problem in distributed search is to resolve possible dependencies among the different search agents. For example, in a CSP constraints may affect the variables in the search spaces of several agents, which makes their search processes inter-dependent. The possible conflicting decisions of the search agents need to be observed and resolved. Also, the efficiency of the search can be affected with coordination.

Development of a distributed search application has an additional task of coordination mechanism design when compared to the development of a centralised search application. The coordination mechanism is dependent on the type of communication available, i.e. shared memory or message passing. The search process may have a varying level of centralised and decentralised coordination. The centralised coordination may have a form of a coordinating agent or global data structure, which is used to guide local search processes of the agents. If the execution order of the local search processes is not restricted the process is called asynchronous search (Yokoo and Ishida 1999). The decentralised coordination may take place via negotiations between the agents. Many distributed search algorithms are extended versions of centralised search algorithms, e.g. distributed backtracking (Yokoo and Ishida 1999).

2.2.4 Distributed planning

Planning may be characterised as one type of problem-solving, in which the problem is to create an action plan in order to take a target system from an initial state to a goal state. Planning can be considered to be one form of path finding search problem. In planning the operators of the search are actions and the states results of performing them (Russel and Norvig 1995). Simple actions can be specified with their preconditions and effects referring to the state of the target system. More complicated action representations are needed e.g. if the time and resources of actions need to be modeled. Planning is often associated with plan execution, with which it may be interleaved. Agents which are expected to act on their environment may use planning as their problem-solving technique.

Development of a planning application involves issues concerning knowledge representation, plan creation method and combination of planning with acting. A central part of knowledge representation in planning is modelling of actions and plans. Representation of plans with several hierarchical levels of abstraction and modelling part of the plans as procedures have been proposed as practical methods for planning. Plans are usually created using search as a problem-solving method. The representation of plans forms the search space. Hierarchical planning that starts from an abstract initial plan and fills it with details is a common arrangement of search in planning. Another often used guideline in planning search is the principle of least-

commitment. According to this commitments to decisions should be made only if necessary. Basic methods for combining planning and acting include conditional planning and re-planning (Russel and Norvig 1995). According to these, a plan can either contain alternative actions or the plan is modified when needed.

Distributed planning is a particular case of planning, in which the plan creation, acting or both are performed by several agents (Durfee 1999, desJardins et al. 1999). A distributed planning problem can be described by stating how the plan creation and execution tasks are decomposed to different agents. The capabilities of agents to act are a common ground for the decomposition of a distributed planning problem. This can be done a priori or during planning. The problem in distributed planning is to resolve possible dependencies among the different planning agents. In principle this can be done either before, after or during local planning inside the agents. During plan execution the actions of different agents need to be synchronised. If re-planning is needed its scope needs to be readjusted via cooperation among the agents.

Similarly to distributed search also the development of distributed planning applications has the additional task of coordination mechanism design when compared to centralised planning applications. Due to the relationship between planning and search also coordination in distributed planning is similar to coordination in distributed search. Similarly to distributed search methods also centralised planning algorithms have been extended to distributed versions, e.g. distributed hierarchical planning (Durfee 1999). However, an important difference between distributed search and distributed planning is that in planning the coordination may be interleaved with both planning and execution. Coordination actions can be regarded as actions to be planned like any other actions.

2.3 Multi-agent system applications in process automation

2.3.1 Overview of multi-agent system applications in automation

There has been a substantial amount of research about several different types of industrial applications of MAS (Parunak 1999). One significant research area of industrial MAS applications has been production control and automation of discrete manufacturing. This topic has been studied particularly within the Holonic Manufacturing Systems (acronym: HMS) consortium (McFarlane and Bussmann 2003). However, research about applications of MAS in process automation has been less extensive than in discrete manufacturing (Chokshi and McFarlane 2002). A possible reason for this is that both the suitability and usefulness of MAS in process automation are maybe not as evident as in discrete manufacturing. Part of the research concerning the applications of MAS in process automation has been conducted within the HMS consortium (Tichy et al. 2002, Maturana et al. 2002, 2003, 2005ab, Chiu et al. 2003), but there has also been other important research efforts (Cockburn and Jennings 1995, Ygge 1998).

In the following chapters the research about the MAS applications in process automation is studied from several viewpoints. Firstly, process automation is characterised as a possible application domain for MAS. The functional requirements, objectives and some other technical developments of process automation are considered from MAS viewpoint. Secondly, there is a description of those research efforts of MAS applications in process automations that may be considered relevant

for this thesis. These applications are studied from the viewpoints of their architecture and functionality. Thirdly, some selected research applications from discrete manufacturing are studied considering their possible impact on applications in process automation. Finally, there is a description of research about using MAS as a systems development method in automation.

2.3.2 Process automation as an application domain

Process automation systems can be characterised as distributed and integrated monitoring, control and coordination systems with partially cyclic and partially event-based operation. The control functions of process automation can be divided into continuous, sequential and batch control (ISA 2000). Particularly the role of continuous control makes process automation different from the automation of discrete manufacturing. In addition to control, process automation has also other functions including performance monitoring, condition monitoring, abnormal situation handling and reporting. In process automation these functions have partially different characteristics than in discrete part manufacturing. Process automation systems are often distributed systems consisting of several controllers running inter-related control applications. There is a need to integrate these systems and coordinate their control operations. This need can be regarded higher in process automation than in discrete manufacturing, because of the significant material flow between the sub-processes in many process automation applications. In addition to this, there is also an increasing need to integrate process automation systems to IT systems, e.g. ERP:s.

The important objectives of process automation have usually been considered to be safety, reliability, quality and efficiency (Rijnsdorp 1991). Flexibility, reconfigurability and responsiveness in a form of recovery operations have not got similar attention than in the research of discrete manufacturing. However, improvement of these properties may be regarded as a feasible research issue also in process automation (Chokshi and McFarlane 2002). Another recently emphasised objective in automation systems is complexity management (Jennings and Bussmann 2003). Automation systems may be regarded complex both in the sense of control and software systems. From this viewpoint facilitation of the engineering of process automation systems can also be considered as an important objective.

Recent technical developments in process automation include e.g. intelligent instrumentation and adoption of new networking and software technologies. Intelligent instruments with their own CPU:s have computing capabilities that could be utilised more extensively. Networking of process automation systems is developing with increasing usage of fieldbuses and connections to information systems via LANs. Object-oriented programming and software component technology has been adopted in implementation of some parts of process automation systems (Kuikka 1999), e.g. in user interfaces. Usage of XML (Karhela 2002), web services (Karhela 2002) and ontologies (Obitko and Marik 2003) has recently been studied as a possible systems integration technology also in the context of process automation. From the software technology viewpoint, process automation systems are becoming increasingly complex, distributed and integrated systems.

2.3.3 Multi-agent system applications in process automation

In the current research of MAS applications in process automation agents have been proposed as a society of one type of intelligent controllers that are expected to operate

at higher, non real-time control levels. Their purpose is to modify the control logic of lower level automation and coordinate these modifications with agent-based coordination methods. It has been assumed that with this role MAS could enhance e.g. the flexibility and responsiveness properties of control functions (Chokshi and McFarlane 2003). The MAS organisations of the proposed applications have usually been hierarchies. A typical organisation has consisted of three levels representing the whole system, subprocesses and equipment (Tichy et al. 2002). Usage of directory systems and Contract Net Protocol as a cooperation mechanism has been proposed as a means to enable changes in the MAS organisations and enhance the reconfigurability of automation systems (Tichy at al. 2002, Chokshi and McFarlane 2003). It has been assumed that suitable hierarchical organisations adequately combine goal-oriented control with the adaptation properties. The agent architectures in the studied applications have been various. Maybe the most advanced ones have been based on the BDI-model (Tichy et al. 2002).

The control functions of process automation to which MAS:s have been proposed to be applied include continuous, discrete and batch control. Depending on the type of control and the approach adopted in the research the general model of a society of intelligent higher-level controllers has appeared in different forms. A common aspect in many studies has been the aim to encapsulate control intelligence and coordination of distributed control operations. Another common feature has been that MAS:s have been considered not only in the normal control state, but also in control during abnormal and planned change situations. However, there have been differences in the methods of control intelligence and coordination.

In continuous control, MAS:s have been proposed primarily for two different purposes. Agents have been proposed to select suitable control algorithms and their parameters for continuous controllers (van Breemen and de Vries 2000ab, 2001). According to this approach agents have been expected to have knowledge about the applicability of different control algorithms. The knowledge is proposed to be represented with rules that map from a situation description to a control algorithm. MAS:s have been proposed as means to coordinate several interdependent continuous controllers. An auction-based negotiation method has been studied as one possible coordination mechanism for this purpose (Ygge and Akkermans 1999). Constraint networks have been proposed as a possible method to represent knowledge about controlled processes (Tyan et al. 1996). They have also been proposed to be used for coordination of multiple continuous controllers.

In discrete control MAS:s have been proposed for planning and execution of control sequences during run-time. A significant research application of this kind is a MAS-based shipboard chilled water control system (acronym: CWS) (Maturana at al. 2002, 2003, 2005ab, Tichy at al. 2002). The purpose of this application is to supervise the chilled water system and change routing of water in it when needed particularly in a fault situation. The agents can create and execute control sequences that change the routing of water though the system. Each agent has planning, plan execution, diagnostics and equipment model modules. The knowledge needed for planning is represented as plan templates. During planning the agents cooperate via negotiations similar to Contract Net. Planning and negotiations together form a search process that creates shared plans of control sequences combining several agents. The search is guided by information about the feasibility and cost of control operations.

In batch control MAS:s have been proposed for planning batch control operations. In a research application, Contract Net is utilised as a negotiation mechanism during the allocation of resources for a batch recipe (Kuikka 1999). In this application, agent-based communication is combined with software component technology. This is achieved with software components that have separate interfaces for agent-based and other type of communication. The decision logic of the agents in this application is represented as decision rules. In another important study (Chokshi 2004) a holonic approach for optimisation of discontinuous control operations of a chemical process was developed. The motivation of this research was to enhance flexibility of production with respect to product and production volume diversity in order to react to changing production conditions. A distributed architecture for holonic process control was presented and argued to be dynamically reconfigurable. Also an interaction model for the holons was designed based on an analogy between holonic process plants and dynamic supply chain networks. Finally, experiments with a distributed algorithm for holonic process optimisation were performed.

MAS applications have been proposed also for other functions of process automation than control functions. Applications for abnormal situation handling and monitoring have been studied. In abnormal situation handling MAS:s have been proposed particularly as a means to modularise a diagnostics system, integrate separate diagnostics systems and coordinate various diagnostics activities. Examples of this approach include ARCHON (Cockburn and Jennings 1995) and MAGIC (Wörn et al. 2002). There are several possibilities for representation of diagnostics knowledge in agents. In the CWS, causal models of process components are used (Chiu et al. 2003). The planning capabilities of the CWS may be used for restoration after diagnosis. Monitoring functions have been experimented with a MAS designed using the BDI-model of agents and the FIPA integration protocols (Pirttioja et al. 2004). In this application plans represent different monitoring tasks that are expected to be easily combined through cooperation among the agents.

2.3.4 Multi-agent system applications in discrete manufacturing

The most important related field to process automation with more extensive research about MAS applications is automation of discrete manufacturing. The current research of MAS applications in this area has to a large extent been conforming to the concept of Holonic Manufacturing Systems (HMS) (Deen 2003). According to this concept manufacturing system control may be designed with so-called holons that form an organisation called holarchy. MAS has been regarded as one possible implementation technology for HMS:s (Marik et al. 2002a, 2005). HMS:s have been proposed to conform to a reference architecture that defines the basic roles of various holon types (Wyns 1999). HMS applications have been studied at several levels of the manufacturing control hierarchy including shop, cell and equipment levels. The concept of HMS does not specify any particular agent architecture. However, deliberative agent behaviour has been proposed in some studies (Heikkilä et al. 1999). The expected utility of HMS has typically been enhanced reconfigurability and adaptation properties (McFarlane and Bussmann 2003).

The control functions of discrete manufacturing to which MAS:s have been applied in research include e.g. production planning (Pechoucek et al. 2005), scheduling (Walker et al. 2005) and shop-floor control (Arai et al. 2001, Luder 2005). In these applications the task of agents has been planning and execution of production plans at

various levels of production planning and control. Several approaches for performing the production planning and shop-floor control functions within HMS or similar systems have been proposed including negotiation (Parunak 1987, Bussmann and Schild 2001, Fletcher and Brennan 2002), economic models (Adelsberger and Conen 2000) and physically (Deen and Fletcher 2000) or biologically (Vaario and Ueda 1996) inspired models.

An important part in many MAS-based shop-floor control studies has been the handling of abnormal situations. The motivation of these studies has been to increase the responsiveness of manufacturing systems. MAS:s have been proposed for applications in both fault diagnosis and recovery. One possible approach to diagnosis is cooperation between possibly several diagnosis agents and monitoring agents (Heck et al. 1998) Approaches proposed to fault recovery include e.g. re-planning of production plans (Fletcher et al. 2001) and reconfiguration of control application (Brennan at al 2002ab). The recovery mechanisms have been considered as a method to make manufacturing systems more fault tolerant (Duffie et al. 1988, Fletcher and Deen 2001). This is expected to be achieved with a decentralised control system and an at least partially redundant manufacturing system. The task of agents is to modify the production plans in such a way that the faulted parts of the manufacturing system are not needed.

2.3.5 Multi-agent systems as systems development method

In development of automation systems MAS has been proposed to have a meaning as a system integration mechanism. Agents have been proposed as modules of an automation system and MAS communication mechanisms as a means to integrate these modules (Cockburn and Jennings 1995, Jennings et al. 1996, Sanz 2000, Wang and Wang 1997). The system integration with MAS is based on a few assumptions. Agents as autonomous, situated and adaptive modules are assumed to be suitable for decomposition of some parts of an automation system (Jennings and Bussmann 2003). The systems to be integrated are expected to be able to be wrapped inside agents. The agent communications mechanisms are assumed to be versatile enough for at least a part of the communication between the modules of automation. They are also proposed to provide higher-level, task-related abstractions for the messages exchanged in automation systems (Jennings and Bussmann 2003).

In addition to integration of automation systems, MAS has been thought to have a more general meaning as a systems development method for automation. In addition to modularisation and versatile communication mechanisms, there are also other aspects of MAS that have been expected to be useful in development of automation systems. MAS societies with organisational structures have been proposed for organizing automation systems (Jennings & Bussmann 2003). Agent architectures have been studied as possible abstractions for representing part of the application logic in automation systems (Krebsbach and Musliner 1998, Ingrand at al. 1992). The coordination mechanisms of MAS has been proposed for building a sort of self-configuration in automation systems (Vaario and Ueda 1996). With self-configuration it is meant the capability of the automation system to adapt to a new situation via reconfiguring parts of itself, e.g. its application logic (Brennan et al. 2001). All the mentioned aspects together are assumed to help managing the complexity of automation systems (Jennings & Bussmann 2003).

2.4 Qualitative reasoning

Qualitative reasoning is a topic in artificial intelligence that studies qualitative models of continuous phenomena and inferences enabled by these representations (Forbus 1996). The essential questions of this topic are, what kinds of qualitative models are useful and which inferences are possible based on them. Qualitative modelling of physical processes may be regarded as a complementing method of modelling with different objectives as compared to quantitative methods e.g. differential equations. Qualitative modelling aims to enable modelling of systems also with incomplete and imprecise information, when creation of quantitative models is not feasible. This is expected to lower the threshold of creating useful models. The reasoning based on the qualitative models is expected to enable easier exploration and interpretation of possible behaviours of target systems. In this way they could give an overview of the behaviour of the target system, while quantitative models are needed for more detailed information.

Several methods for representing various aspects of continuous phenomena have been studied within the research of qualitative reasoning (Forbus 1996). Essential aspects of modelling in qualitative reasoning include e.g. quantity, mathematical relations, time and state. A basic method for representing quantity is so-called sign algebra (deKleer and Brown 1984, Iwasaki 1997). More advanced methods include e.g. usage of intervals and order of magnitude representations. In the representation of mathematical relations qualitative arithmetic has a central role. Applications of qualitative reasoning are proposed to be constructed via combining parts of so-called domain theories that model selected aspects of an application domain based on defined modelling assumptions (Forbus 1996). The results of an application of qualitative reasoning may contain e.g. description of the target system in terms of its qualitative states and their changes over time.

Qualitative representations enable inferences about the target processes for various purposes. Typical usage of qualitative inference include e.g. diagnosis, simulation, comparative analysis, data interpretation and planning (Forbus 1996). Qualitative diagnostics reasoning is usually based on modeled causalities which is also expected to enable explanation of faults. Qualitative simulation and comparative analysis contains exploration of qualitative state space of the target system which might be useful e.g. in diagnosis and planning applications. State spaces are proposed to be used for data interpretation application by interpreting measurements as a sequence of qualitative states.

2.5 Discussion

The methodology of MAS is discussed in here as a methodology to be applied in process automation and the presented MAS applications in process automation as the state-of-the-art in this research topic. The possible benefits and problems of applying MAS to process automation are characterised. The value and limitations of the research results already achieved in MAS applications to process automation is assessed. Finally, essential open questions are outlined.

The methodology of MAS has properties that seem to suggest its applicability to at least some functions of process automation. The situatedness and goal-orientedness properties of MAS appear to match with the requirements of process automation

relatively directly. Concerning the communication, coordination and adaptation properties the situation is more complicated. It is not clear to which extent FIPA-type of communication and MAS-based coordination mechanisms can fulfil the requirements of process automation. The MAS architectures are likely to be applicable in process automation, but with the problem-solving methods the situation is more unclear. It is not clear to which extent the distributed versions of search and planning methods are able to satisfy the requirements of process automation. Finally, whereas agents with autonomy are argued to be useful abstractions for managing complexity of software systems, it is not self-evident if they are similarly applicable to management of complexity in process automation.

The results in the research of MAS applications in process automation provide some insight about feasible and useful ways of applying MAS in process automation. The utilisation of MAS at higher, supervisory control levels and hierarchical organisations of MAS applications may be argued to be reasonable matches between the properties of MAS and the requirements of process automation. In the current research MAS:s have already been applied to several relevant control types of process automation (Tichy at al. 2002, Ygge and Akkermans 1999, Kuikka 1999, Chokshi 2004) and also to other functions (Wörn et al. 2002). The MAS-based problem-solving methods proposed for different functions appear adequate for their purposes at least in the scope of the presented studies. The expected positive effect on the adaptation properties of automation, particularly reconfigurability, seems justified, but its significance has not been evaluated comprehensively. There have also been other important limitations in the reported research. The designs of MAS:s for different functions have been different. Research results concerning the performance of the presented methods in control operations have been limited. Validation of the MAS-based control systems have been found difficult (Hall et al. 2005).

The state-of-the art in the research about MAS applications in process automation leaves some essential open questions. The research has focused more on control functions concerning the functional role of MAS in process automation. Other functions, e.g. monitoring and information access have got less attention (Pirttioja et al. 2004). There are also open questions relating to the architecture of MAS applications in process automation. One such question is if there could be a common programming model for MAS applications in various functions of process automation including different types of control. A particular question in this context is if both sequential and continuous control functions could be programmed with the same model (Seilonen et al. 2004). There are also important open questions concerning the MAS-based problem-solving methods (Seilonen et al. 2003b). To a large extent it is an open question under which conditions the MAS-based problem-solving methods enable implementation of reliable control operations. Finally, the effect of MAS applications on the properties of process automation is still unclear.

3 Agent platform for process automation

3.1 Introduction

This chapter presents the development of the agent platform for process automation. This development is based on the related research and methodologies described in Chapter 2. It is also a basis for the applications specified in the subsequent chapters. The target of the development is an agent platform for the process automation application domain. This platform enables implementation of working applications and can be argued to enhance some properties of process automation systems. The platform is particularly required to support design of applications for both sequential and supervisory continuous control.

The development work described in this chapter contains a specification of a suitable architecture for the agent platform and an experimental implementation of it in a laboratory test environment. The specification contains models of the agent platform both at agent society and agent levels. The purpose of the experimental implementation is to verify the feasibility of the specification of the agent platform. The implementation contains a simple laboratory test process, an automation system, an agent platform and a test application. The end of the chapter presents a discussion about the properties of the developed agent platform for process automation and its relations to the previously published approaches.

3.2 Specification of the agent platform

3.2.1 Agent society model

The society model of the agent platform for process automation describes the role of the agents as a part of an extended process automation system, principles of the agent organisation and non application specific aspects of its operation. These aspects include the basic principles of communication and coordination among the agents. An agent model complementing the agent society model is depicted in the following chapter (Chapter 3.2.2). Application-specific features of the agents are described later in chapters 4 and 5.

The agent platform for process automation together with the applications built with it operate as a higher-level automation layer on top of an ordinary process automation system as illustrated in Figure 3.1. The functional role of the agent platform is to run supervisory control applications that make decisions about higher-level control operations and execute them by changing the control parameters of the lower-level automation system. The higher-level control operations of the agent applications may relate to continuous control, e.g. changing the set-points of lower level controllers, or they can be control sequences consisting of actions of lower level automation, e.g. in batch control applications. Real-time cyclic control and other time critical control operations are run in the lower-level automation system. Both the agent-based layer and the ordinary automation system may be distributed systems. The agents may also provide services for external clients and can be connected to neighbouring automation systems, e.g. in other process areas.

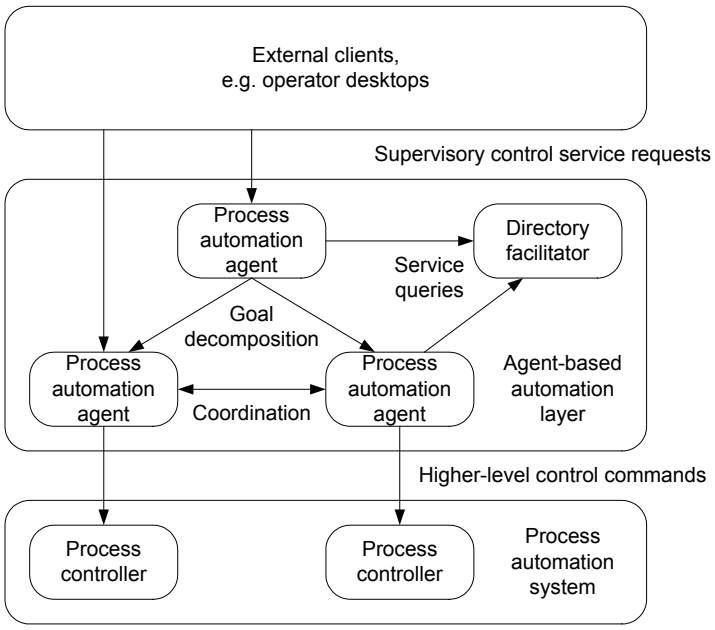


Figure 3.1 Relationships of the agent-based automation layer with other systems.

The agent platform consists of process automation agents which conform to the rules of the agent society. All process automation agents are functionally similar. They can be characterised as semi-autonomous, i.e. they can carry out their activities independently but they also cooperate with other agents. They communicate according to the FIPA-standard (FIPA 2005) using predefined communicative acts and interaction protocols. The agent society has a hierarchical organisation based on authority relations (see Figure 3.2). Agents at the lower-levels of the organisation typically supervise parts of the controlled process and its automation system as their areas of responsibility (see Figure 3.1). Only these agents control the lower level automation system directly. The areas of responsibility may map e.g. to the physical or functional division of the process. The higher-level agents supervise larger areas of the controlled process indirectly via their subordinates. In addition to the process automation agents, the agent society contains a directory facilitator agent (acronym: DF). This agent maintains a registry about other agents and their services. Except for the directory facilitator, the agent society does not have any centralised data storage. Each agent manages its data locally. They can also access data sources outside the agent platform, e.g. in the lower level automation.

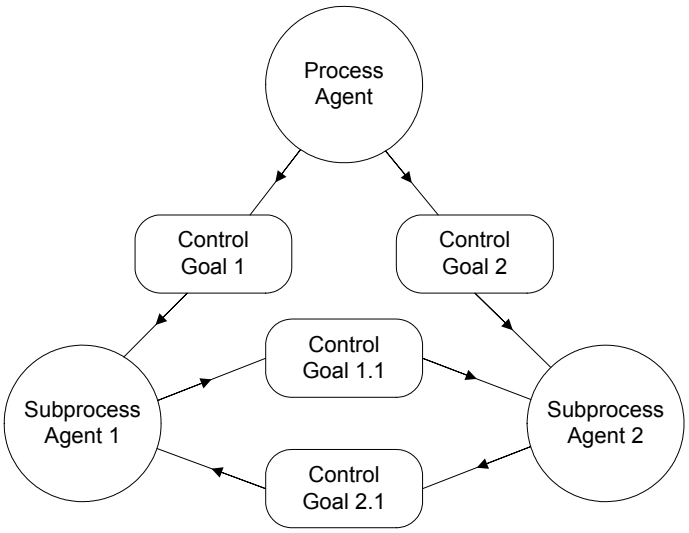


Figure 3.2 Organisation of process automation agents represented as Tropos actor diagram.

The process automation agents share a common ontology that specifies the concepts that the agents need in their communication. The agents conform to the agent management ontology as defined in the FIPA standard (FIPA 2004). The concepts of this ontology are needed when accessing information about agent services registered to the directory facilitator. The agents also share another ontology that specifies the concepts they need when cooperatively performing process automation operations. This ontology is illustrated in Figure 3.3. The main concepts in this ontology are agent relation, goal, contract and process variable. Agent relations are used to express the organisational supervisor vs. subordinate relations between the agents. Information about goals and contracts is exchanged between the agents during cooperative planning and execution of automation operations. Goals may refer to process variables that are partially shared knowledge among the agents.

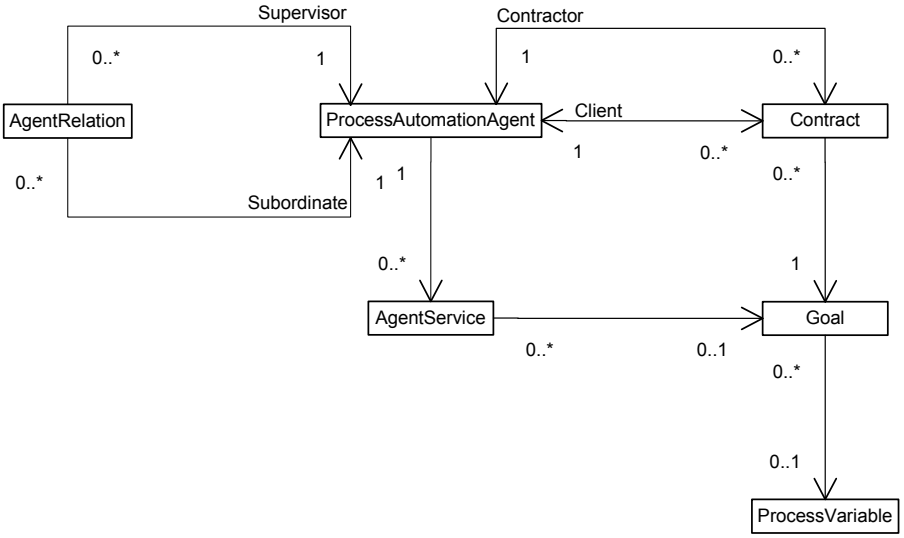


Figure 3.3 Shared ontology of process automation agents represented as an UML class diagram.

The operation of the process automation agent society as a higher-level supervisory automation layer can be decomposed into a set of activities among the agents. The

main activities of the agents include monitoring the lower-level automation system, processing of information queries, registration and search of agent services, planning of control operations and execution of planned operations. The agents perform these activities in a distributed and parallel fashion. While monitoring is an ongoing activity, query processing, planning and plan execution are performed when needed. The initiative for planning and plan execution can originate either from monitoring or from an external client. The monitoring and query processing activities are not studied any further in this thesis. They are a part of another research effort (Pirttioja at al. 2004). Details of the other activities are explained below.

During agent service registration and search the process control agents utilise a directory facilitator as defined in the FIPA agent management specification (FIPA 2004). The interaction between process automation agents and a directory facilitator is illustrated in Figure 3.4. During start-up the process automation agents register their services and organisational position to the directory facilitator according to the concepts of the shared ontology. During planning they can then identify other agents based on this information. They can for example search for agents providing services relating to a particular goal, an agent that can affect a certain process variable or an agent in a particular position in the agent organisation.

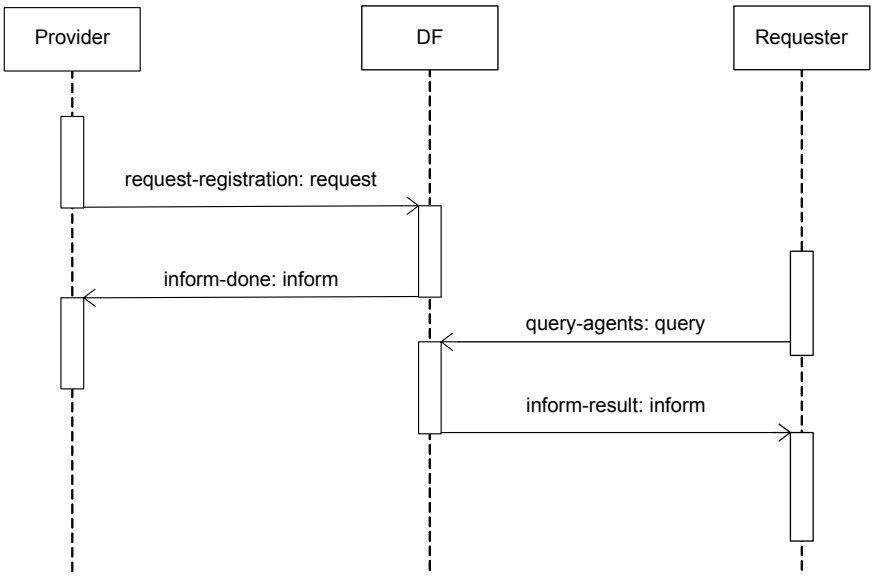


Figure 3.4 Interaction between process automation agents and a directory facilitator during service registration and search represented as an AUML sequence diagram (Odell et al. 2001).

During the planning of control operations the process automation agents utilise the FIPA Contract Net interaction protocol (FIPA 2002d). The agents use this protocol in order to make contracts about fulfilment of process control goals. The negotiation protocol is illustrated in Figure 3.5. The call-for-proposal message specifies a goal that the initiator wishes to be fulfilled. The proposal message depicts to which extent a participant is willing to commit to the goal. When the initiator receives a proposal from a partner a tentative contract is created between the partners. The initiator can send an accept-proposal message to one or more participants depending on the type of the goal under negotiation. It may contain full or partial acceptance of the proposal. The accept-proposal message may also specify if the participant should execute the actions associated to the contract immediately or only after a further request message.

The negotiation process may be carried out through both the vertical and horizontal cooperation channels. On one hand, supervisor agents can assign sub-goals of their own goals to their subordinates. On the other hand, peer agents can negotiate with their peers in order to handle interrelations between their goals. The negotiations may be chained, i.e. one negotiation is started because of another one. In chained negotiations accept-proposal messages are sent only after the agent who started the entire negotiation process observes that it can fulfil all of its goals. After that all involved agents receive accept messages via the chain of related negotiations.

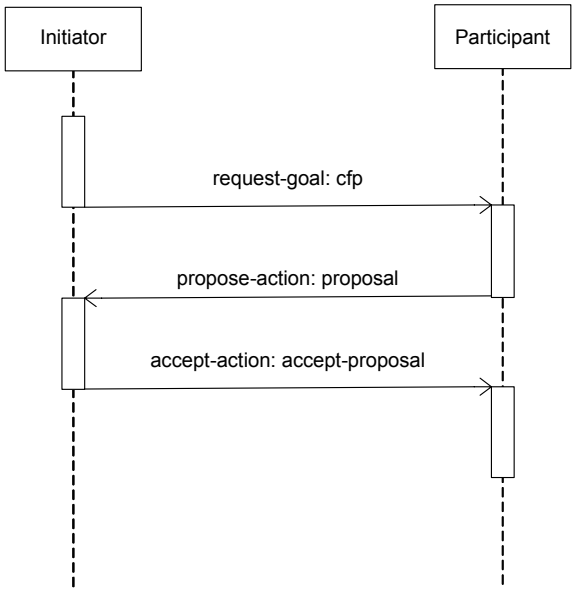


Figure 3.5 Interaction between two process automation agents during planning of control operations represented as an AUML sequence diagram.

During the execution of control plans the process automation agents utilise the FIPA Request interaction protocol (FIPA 2002e). The agents use this protocol for requesting execution of the plans associated with previously made contracts. As in planning, also this protocol can be used through both the vertical and horizontal cooperation channels. The interaction protocol is illustrated in Figure 3.6. A request message specifies a contract to be fulfilled. An inform message describes the result of the plan execution.

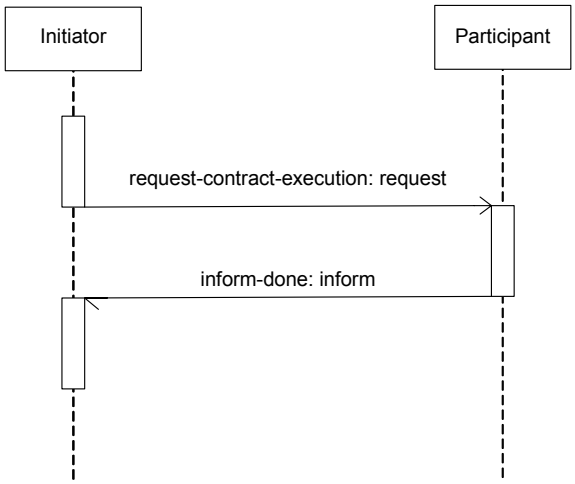


Figure 3.6 Interaction between two process automation agents during control plan execution represented as an AUML sequence diagram.

3.2.2 Agent model

The agent model of a process automation agent specifies the internal modules of an agent and their operation. The specification in here only covers those aspects that are not specific to any particular application. Specifications for two different experimental applications are presented in Chapters 4 and 5. The agent model is designed using the BDI (Georgeff at al. 1999) and PEM (Heikkilä at al. 1999) models as references. The type of the BDI-model applied is the one utilising procedural plans. The mentioned agent models are goal-oriented and fit to the specified society model of the process automation agents. The specification shows how a similar goal-oriented agent model can be applied to the internal design of these agents.

In addition to the goal-oriented agent model, the process automation agents need to conform to the agent model of some FIPA-compliant generic agent platform. For the agent model of the process automation agents the compatibility with FIPA means that the process automation agents conform to the FIPA specifications of abstract architecture (FIPA 2002a), agent management (FIPA 2004) and communication (FIPA 2002c). For the internal architecture of the process automation agents there are several options depending on the underlying generic agent platform. It is expected that the following agent model can be designed on top of several different internal agent architectures e.g. task-model of FIPA-OS (FIPA-OS 2005), behaviour-model of JADE (JADE 2005) and BDI-model of Jadex (Jadex 2005).

A process automation agent consists of modules that can be categorised as operational and modelling modules and run-time data structures. The internal structure of a process automation agent is illustrated in Figure 3.7. The operational modules take care of the activities of a process automation agent. They use the models and update the run-time data structures. The modules relating to the BDI-model are the Planner, Executor, plan library, goals, plans and actions. They implement agent architecture similar to JAM (Huber 2000). Agent relations and contracts are used for enabling cooperation among the agents. The process model is required for control operations. The various parts of a process automation agent are described in more detail below.

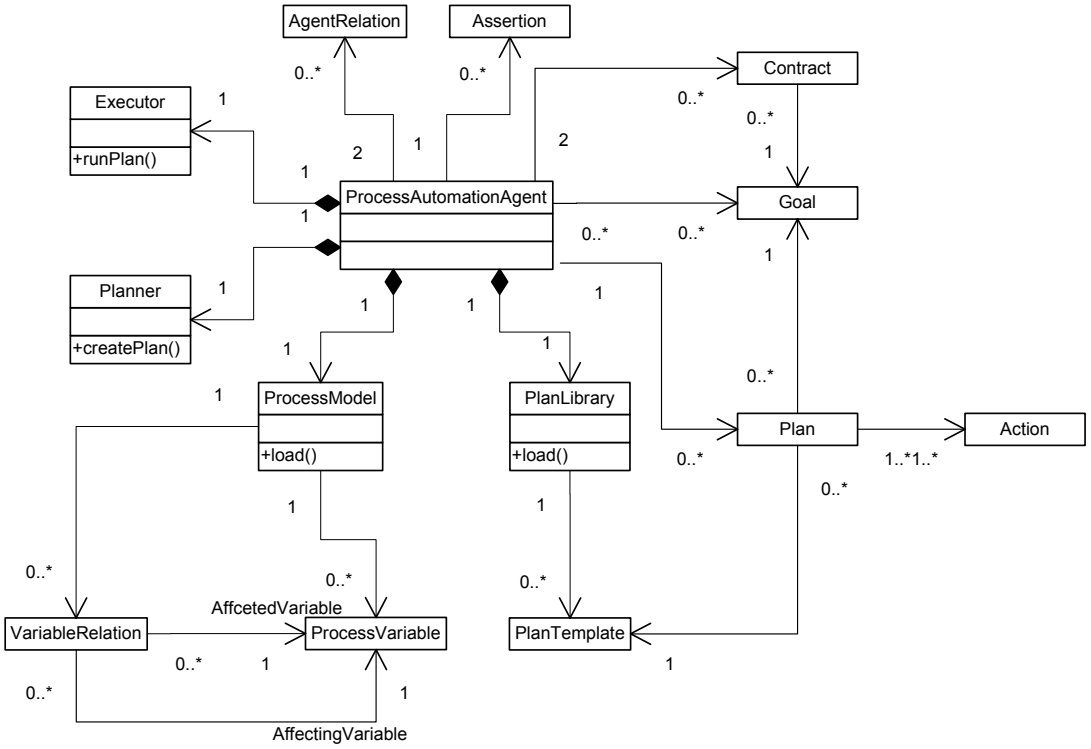


Figure 3.7 Modules of a process automation agent represented as an UML class diagram.

The Planner module contains data structures and functions needed for planning activities of a process automation agent. Because planning in a process automation agent is an activity both at the agent society and agent levels, the planner is designed to take into account both of them (see Figure 3.5 and Figure 3.9). The planner module integrates local planning of an agent with Contract Net type of negotiation. The main functions of the planner include a planning algorithm and decision-making functions needed during negotiation. The planning algorithm takes the goals, the process model and the plan library as input and creates the plans. Decision-making functions for creating bid, proposal and answer messages of the Contract Net are needed during negotiation. The result of negotiation is a contract. The interplay between the planning algorithm and negotiation can bidirectional. The planning algorithm initiates negotiation if it needs to request goals from other agents. The opposite situation occurs when negotiation initiates planning because of a requested goal.

The Executor module contains data structures and functions needed for execution of the control plans. Because also the plan execution in a process automation agent is done combining both the agent society and agent levels, the executor is designed for both of them (see Figure 3.6 and Figure 3.10). It integrates local plan execution of an agent with coordination via FIPA Request interaction protocol. The local plan execution takes run-time plans as input and executes control actions via the process model. Coordination is based on contracts. Likewise in planning the interaction between the local plan execution and coordination may be bidirectional. An agent can execute a run-time plan as part of its own plan execution process or because of a request from another agent.

The agent organisation model describes the knowledge of a process automation agent about its relations to the other members of the agent society. The model is needed to

express the hierarchical agent organisation. Each process automation agent knows its direct supervisor and subordinates. This information is configured during agent application development and registered to the directory facilitator during agent startup.

The process model describes the knowledge of a process automation agent about the controlled process. A process automation agent can have knowledge about the existence of process variables, their measured values and relations between the variables. Each process automation agent can be configured with zero or more process variables that it can either only measure or also control. A control variable may be controlled only by one agent whereas measurements can be shared among several agents. In addition to the existence and value information of variables a process automation agent can also be configured with knowledge about the relations between the variables. At least qualitative knowledge about the relations between the process variables is needed for facilitating coordination among the process control agents. An agent may make inferences by the qualitative process model about which process variables controlled by other agents affect its variables, and initiate a negotiation with them. Also the process model is configured during agent application development. Some of the variables of an agent can be registered to the directory facilitator as public variables. In this case also other agents can know about their existence and initiate negotiations about their control.

The plan library contains plans that a process automation agent can use during planning in order to create run-time plans. Each process automation agent is configured during agent application development with a set of plans that it needs in order to be able to plan its control operations. The plans are expressed with a predefined plan language. The plans are based on the principles of procedural reasoning (Ingrand at al. 1992). A plan associates a goal with sub-goals, a procedure and preconditions. The procedure can contain both process control and negotiation actions. A process control agent can register process control services relating to the goals of its plans to the directory facilitator and thus make them available also to other agents.

The important run-time data structures of a process automation agent include goals, run-time plans and contracts. The purpose of these data structures is to hold information about which goals an agent is trying to fulfil, and which plans and contracts it has made in order to do so. Goals and contracts are modelled according to the shared ontology of the process automation agent society. Run-time plans are internal data structures of each agent. An agent creates them based on its goals and the plan library with its planner module. Assertions are another important set of run-time data structures. Their purpose is to store necessary information between separate planning sessions of an agent. Assertions are useful e.g. in cyclic control operations for storing past values of control parameters.

The main activities of a process automation agent are the agent level counterparts of the activities of the agent society, i.e. monitoring the lower-level automation system, service request intake from external clients, processing of information queries, registration and location of agent services, planning of control operations and execution of planned operations. The role of the agent modules in the implementation of these activities is illustrated in Figure 3.8. The planning and plan execution

activities are described with more detail below. At the agent level these activities are a combination of local activities of an agent and coordination of them with other agents.

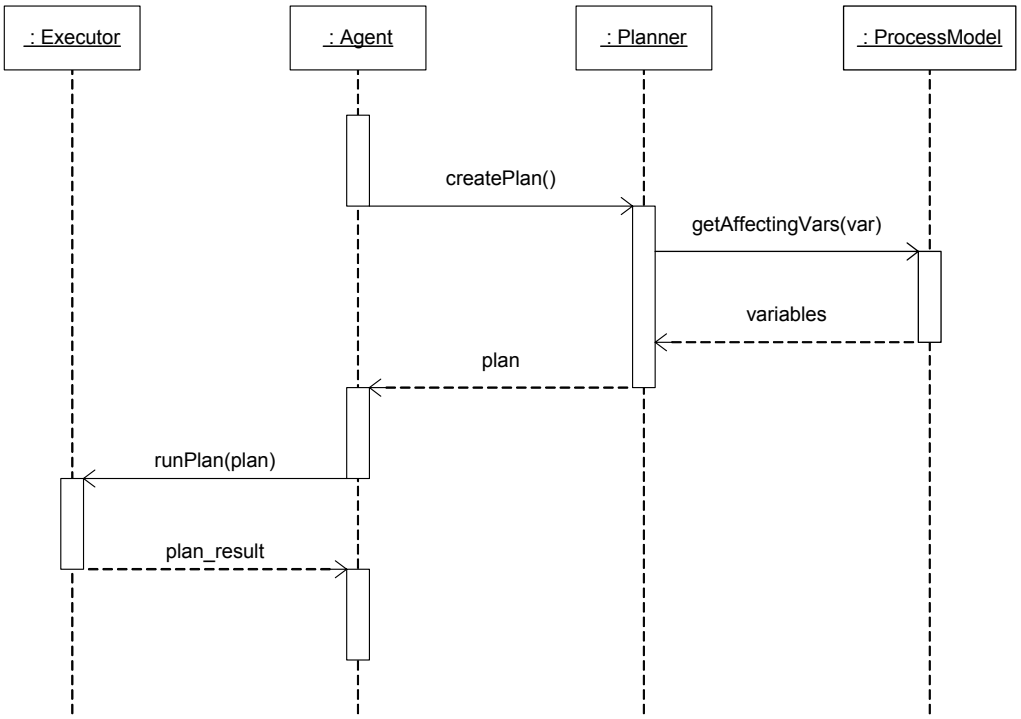


Figure 3.8 Interaction between operational modules of a process automation agent represented as an UML sequence diagram.

The purpose of the planning activity is to create plans and associated contracts that can fulfil the current goals of an agent if possible. The planning activity is illustrated in Figure 3.9. The starting point of planning is a new goal request and the set of current plans and contracts. The planner selects a suitable set of plans from the plan library and the planning algorithm checks if the new set of goals can be fulfilled with these. The planning process contains decomposition of goals, definition of local action plans and negotiation with other agents about contracting goals as specified in the agent society model. During planning the process model is used to access values of process variables and make inferences about qualitative relations between them. An agent can observe interrelations to variables of other agents and initiate negotiations. The agent organisation model can also be used for locating suitable partners to negotiate with. In the end a new set of plans and contracts is created or a failure to fulfil the goals is signalled. The planning activity can appear in different forms depending on the complexity of the planning task and the need to coordinate it with other agents. It may require planning of a sequence of plans or it might be a simple, reactive choice of one plan. Both planned sequences of actions and reactive actions may need to be coordinated with other agents depending on their effect on the target process.

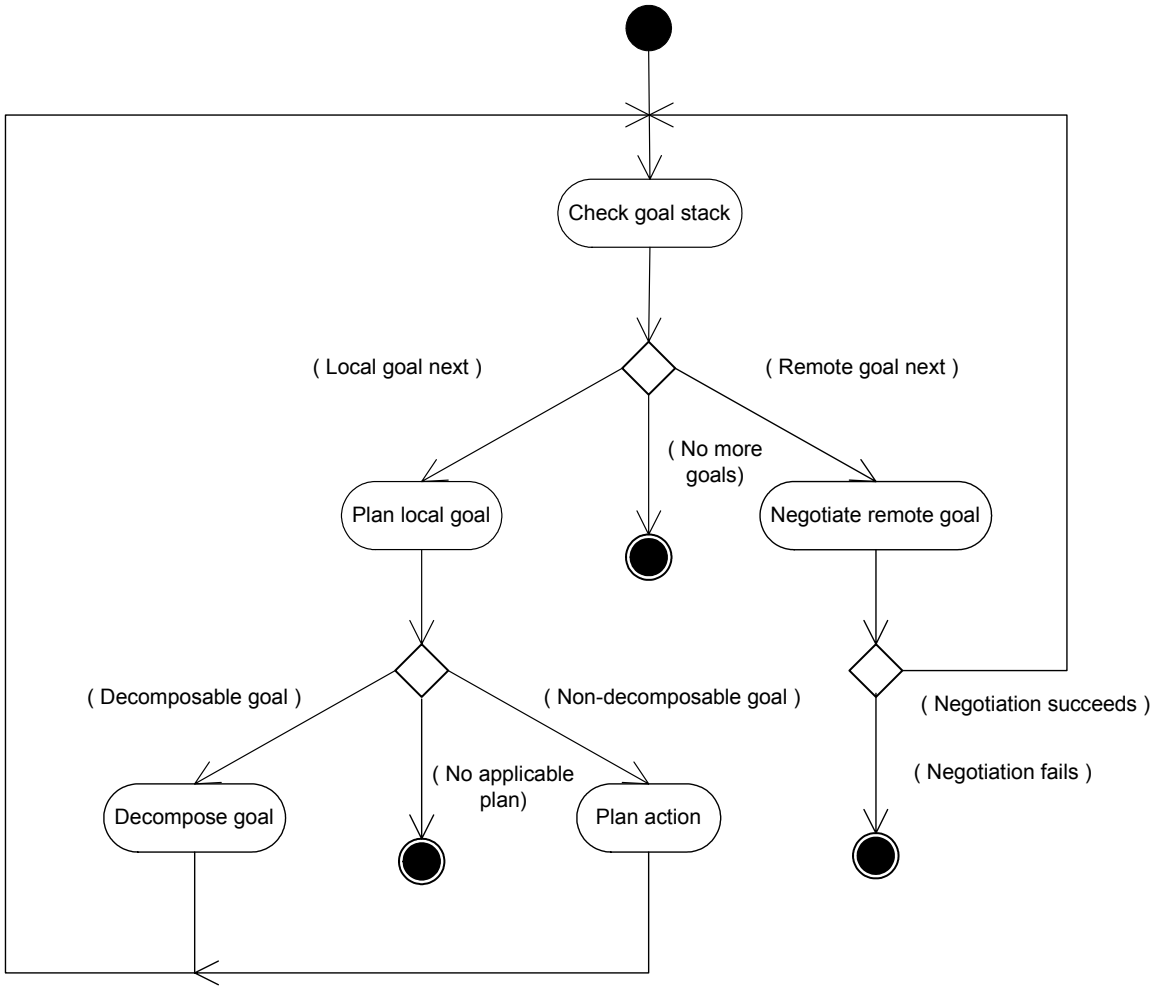


Figure 3.9 Planning activity of a process automation agent represented as an UML activity diagram.

The purpose of the plan execution activity is to execute control operations of plans and communicate information about this to other agents when needed. The plan execution activity is illustrated in Figure 3.10. The starting point of plan execution is a request to execute a previously planned control plan. After this the agent starts executing the actions defined in the requested plan step by step. Process control actions are executed locally within each agent. Coordination actions are executed as requests to other agents to fulfil the contracts made during planning.

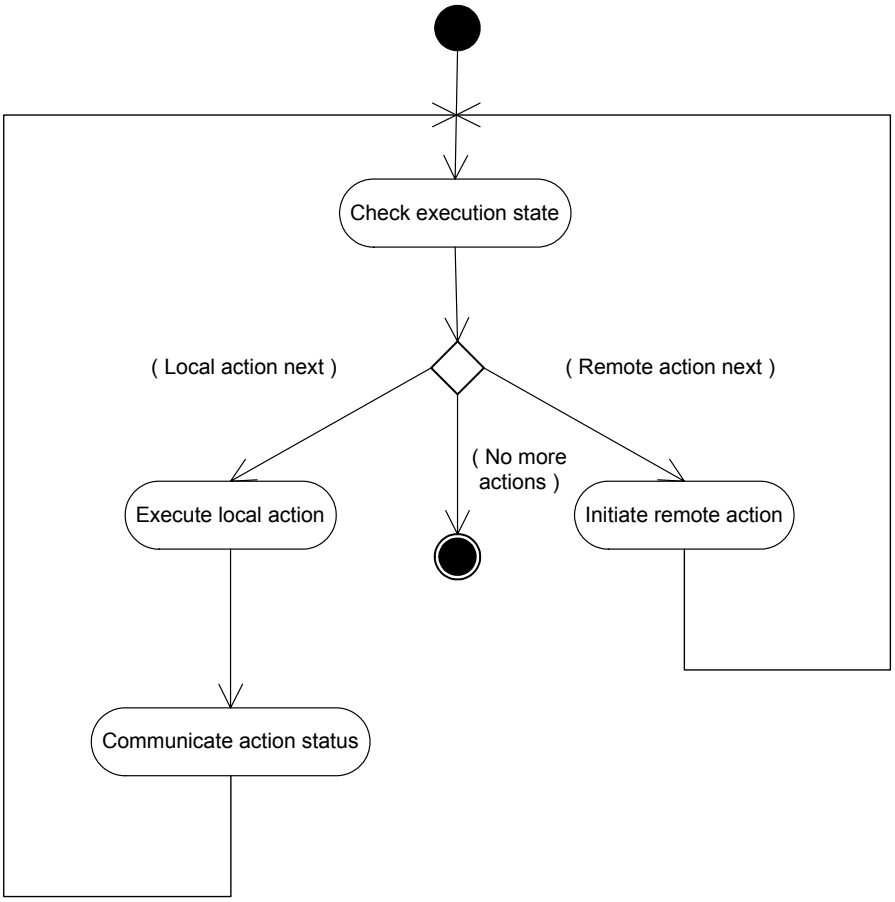


Figure 3.10 Plan execution activity of a process automation agent represented as an UML activity diagram.

The internal structure of a process automation agent is organised according to a layered architecture illustrated in Table 3.1. The lowest layer in this architecture is a generic agent platform that provides a run-time environment and general agent functions. The middle layer is the process automation agent platform. This layer consists of the operational modules, run-time data structures and models of a process automation agent. The highest layer of the architecture is an application implemented with the process automation agent architecture. At this layer the models are configured for a specific process and application.

Table 3.1 Layer model of process automation agents.

Agent application
- Plans
- Configuration of the process model
- Configuration of the agent organisation
Agent platform for process automation
- Operational modules Planner and Executor
- Run-time data structures of planning, execution and cooperation
- Process model and agent organisation model
Generic agent platform
- Agent management
- Agent communication
- Agent tasks

3.3 Test environment for the agent platform

3.3.1 Test process and automation system

A laboratory test environment has been used as a test bed for an experimental implementation of the agent platform for process automation (Pirttioja 2002, Chakraborty 2003, Fajt 2003, Seilonen 2002ab, 2003abc, 2005). The purpose of the test environment in this research is to verify the feasibility of the implementation of the agent platform and enable experimentation with its applications. The test environment consists of several parts including a test process, instrumentation, an automation system, a control application, an operator user interface, an agent platform and two agent applications. The experimental agent applications are described more closely in chapters 4 and 5. The other parts of the test environment are depicted below.

The test process is a small-scale water temperature control process illustrated in Figure 3.11. The main components of the process include a tank, a pump and pipes connecting these. Water level and temperature in the tank are controlled with five control valves (see Figure 3.12). Process flow is imitated by circulating water with the help of the pump. The instrumentation of the process also includes several temperature sensors and a pressure sensor, a flow sensor and four magnetic valves.

The control system of the test process runs three control loops. One is a water level control loop for stabilising the water level in the tank. The purpose of the other two control loops is to control the water temperature at the upper and the lower parts of the tank. The objective of the water temperature control is to keep the temperature at given values in both parts of the tank. These values might be different which creates a simple temperature profile. The temperature is calculated as the average of two separate measurements at the both parts. Separate PID controllers are used in all three control loops (Pirttioja 2002). However, the temperature control loops are interrelated trough the water flow in the process. The water temperature at the upper part of the tank has a strong effect on the temperature at the lower part through the water flow in

the tank. A weaker reverse effect is caused by the water flow through the pipes. The temperature control as a whole can be regarded as a MIMO control problem, in which the inputs of the controller are the temperature setpoints and the outputs of the controlled system are the temperature measurements. The setpoints of the pump and the valves are the control variables.

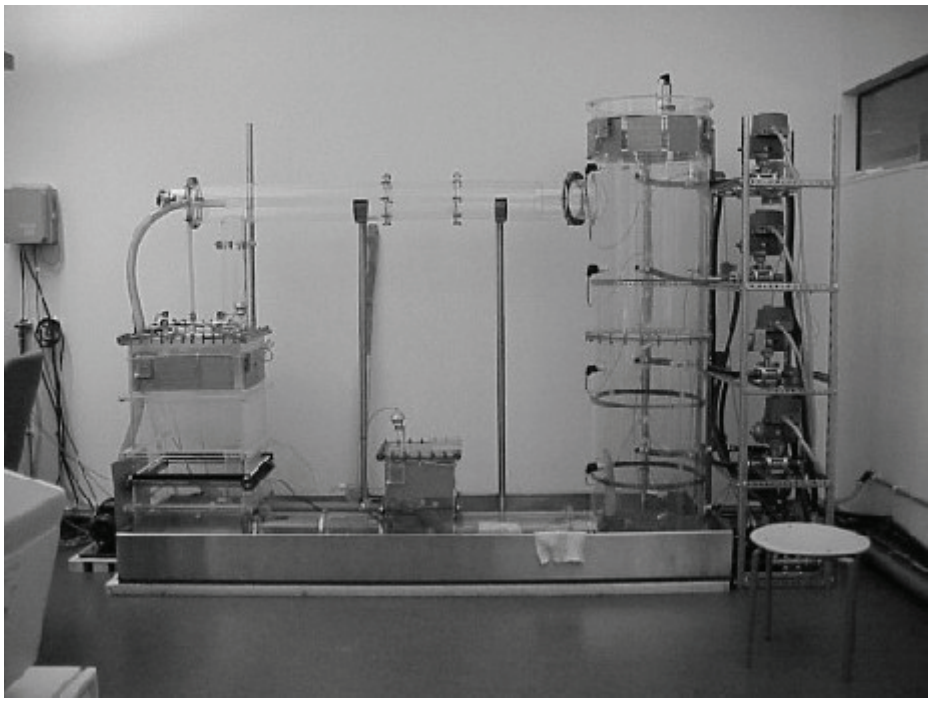


Figure 3.11 Physical layout of the test process.

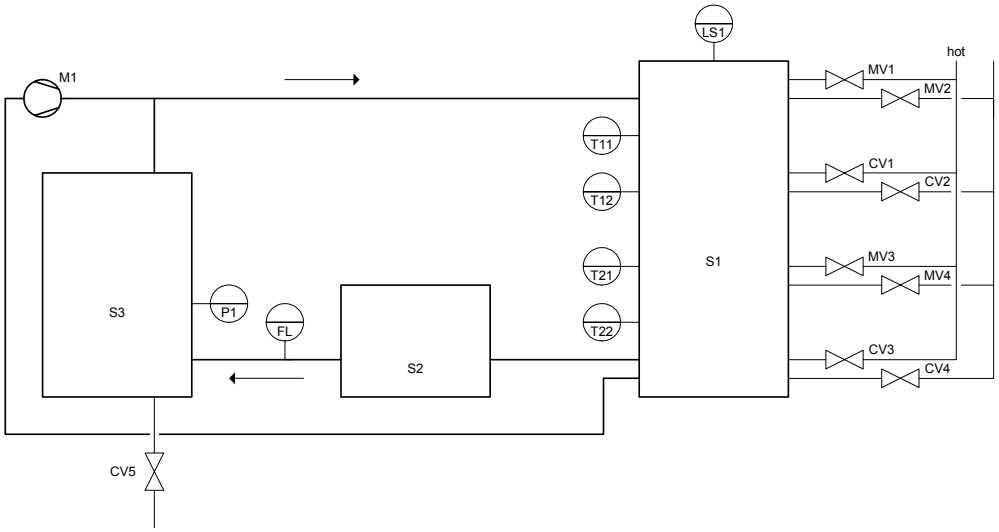


Figure 3.12 PI-diagram of the test process.

The automation system in the test environment consists of a Smar DFi 302 process controller (Smar 2002), a control application and an operator user interface as illustrated in Figure 3.13. The instrumentation is partly connected to the process controller by a Foundation Fieldbus technology-based fieldbus and partly by a distributed I/O system. The control application implementing the control loops is partly run at the controller and partly at the control valves. The operator user interface was built as an application with iFix (GE Fanuc Automation 2005) for monitoring the

process and controlling its instrumentation (see Figure 3.14). It is connected to the controller via an OPC server.

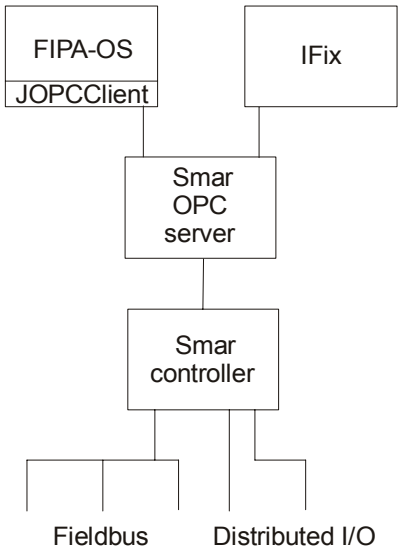


Figure 3.13 Automation system of the test process.

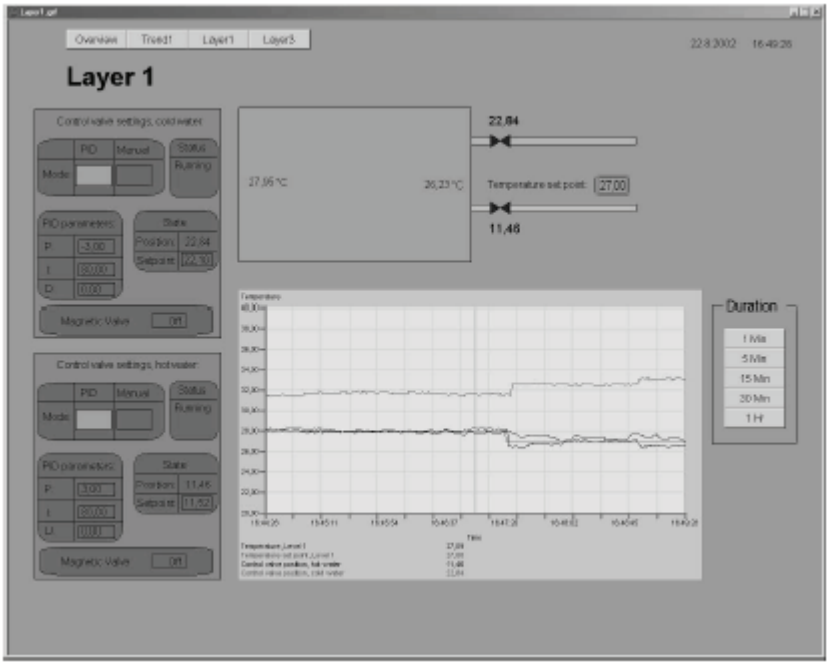


Figure 3.14 Screenshot of the operator user interface of the test automation system (Pirttioja 2002).

3.3.2 Test agent platform and application

An experimental implementation of the process automation agent platform specified in Chapters 3.2 and 3.3 was built as a part of the test environment. Two experimental agent applications were developed with it. The platform is designed to be generic within the process automation application domain whereas the applications are configured to the specific test environment.

The process automation agent platform was implemented with several software tools based on Java programming language. The most important ones of them were FIPA-OS (FIPA-OS 2005) and JAM (Huber 2000). FIPA-OS is a FIPA-standard compliant agent platform. FIPA-OS was used as a generic agent platform providing FIPA-compliant agent communication and agent management services including a directory facilitator. FIPA-OS also provided a task-model-based internal agent architecture for the process automation agents. JAM is a hybrid agent architecture built upon the ideas of procedural reasoning (Ingrand at al. 1992) and BDI agent model (Georgeff at al. 1999). JAM was used to implement the planning capabilities of the process automation agents. The agent models of FIPA-OS and JAM were integrated in the implementation. This enabled application development for the process automation agents with JAM. According to JAM, applications are defined using a special plan representation language (Huber 1999) and interpreted by an agent interpreter during run-time. Pseudocodes of some JAM plans developed as applications of the test agent platform are presented in subsequent chapters (Chapter 4 and Chapter 5). Other software tools used in the implementation include Jess (Jess 2005) and JOPCClient (OPI 2005). Jess was used for implementation of a simple process monitoring module for helping in the testing of the experimental applications. JOPCClient was used for implementation of a connection to the automation system from the agent platform via an OPC server (Pirttioja 2002).

The agent society in the test applications has five process automation agents and a directory facilitator as illustrated in Figure 3.15. The agents follow the rules of a process automation agent society as defined in Chapter 3.2. The topmost agent is Process Agent which supervises the whole test process indirectly via its subordinates. Tank Agent and Pump Agent supervise their physical sub-processes respectively. Upper Part Agent and Lower Part Agent have the temperature control loops as their areas of responsibility. All the above-mentioned agents register their capabilities to one Directory Facilitator Agent present in the test system. The qualitative process model of the test process used by the agents is illustrated in Equations 3.1, 3.2 and 3.3. This model captures the effect of temperature T1 and water flow to temperature T2. The reverse effect from T2 to T1 is ignored. The configuration of the agents is presented in Table 3.1.

$$M+(T_2,\ T_1) \quad [3.1]$$

$$T_1 > T_2: M+(T_2,\ V) \quad [3.2]$$

$$T_1 < T_2: M-(T_2,\ V) \quad [3.3]$$

T_1:	Water temperature at upper part of the tank, average of T_{11} and T_{12}
T_2:	Water temperature at lower part of the tank, average of T_{21} and T_{22}
V:	Water flow through the tank
M+(y, x):	y is monotonically increasing function of x
M-(y, x):	y is monotonically decreasing function of x

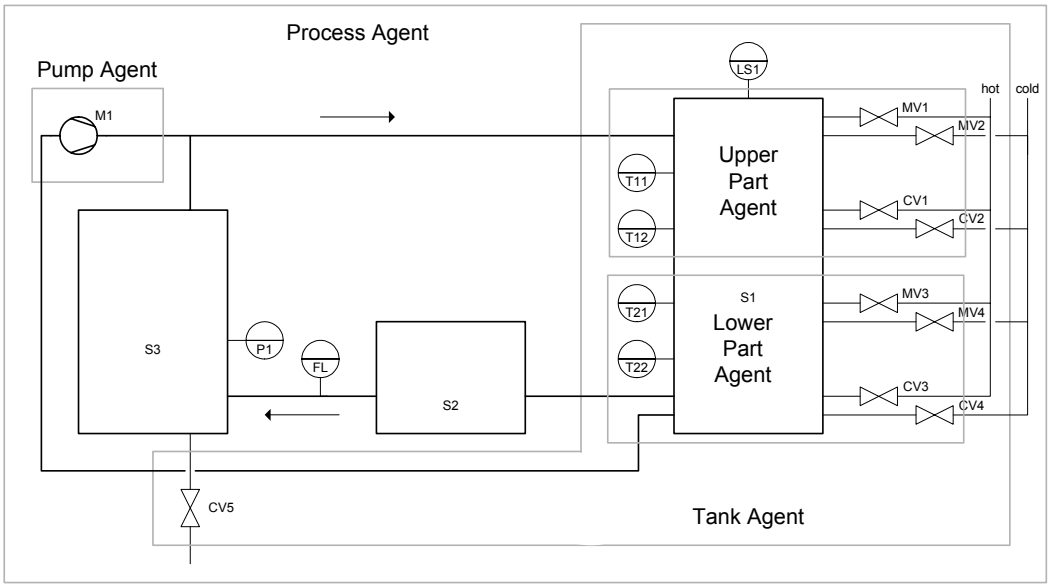


Figure 3.15 Agents in the test applications.

Table 3.2 Configuration of the agents in the test applications.

Name	Parent	Measurements	Controls	Registered goal names
Process Agent	none	none	none	startup, shutdown
Tank Agent	Process Agent	LS1	CV5	startup, filled, shutdown
Pump Agent	Process Agent	none	M1	startup, flow, shutdown, recovery _help
Upper Part Agent	Tank Agent	T11, T12	CV1, CV2, MV1, MV2	temperature_control, fault_recovery, recovery_help
Lower Part Agent	Tank Agent	T21, T22	CV3, CV4, MV3, MV4	temperature_control, filling, fault_recovery, recovery_help

3.4 Discussion

The specification of the process automation agent platform is assessed here by comparing it to the related research of agent-based process automation systems and discussing its possible effect on the properties of process automation. Similarities and differences to the references are pointed out. Justifications are presented for the essential design decisions taken in the specification. The effect of these choices to the automation system properties is discussed. Finally, conclusions are drawn based on this study and open questions concerning the research topic are outlined.

3.4.1 Design choices in the specification

The society model of the agent platform for process automation has many similarities with the HMS-based approaches in automation, particularly the CWS (Tichy at al. 2002, Maturana 2002, 2003) The similarities include the role of agents as a supervisory control system, a hierarchical organisation similar to a holarchy, goal-oriented operation of the agent society, usage of a directory facilitator and conformance to the FIPA-standard. These features were assessed to be suitable based on the earlier research results of MAS applications in automation. However, there are also a few particular aspects in the agent society model of this study. These are aimed either for noting the specific characteristics of process automation or facilitating the requirements relating to the adaptation properties. The agent society model has only one agent type. This design is motivated by its similarity to the architectures of process automation systems. Agents are considered as generic automation system modules analogous to process controllers. In this aspect the presented specification is similar to the CWS but different from many other HMS studies (Wyns 1999, Chokshi 2004). Another important difference to the references is that the agent society model of this research allows coordination via both vertical and horizontal coordination channels. This design results in a system architecture, which is expected to further facilitate the adaptation properties.

The agent model of the agent platform for process automation is quite similar to those HMS models that are also based on BDI-agents or similar approaches (Heikkilä at al. 1999, Tichy at al. 2002, Maturana at al. 2002, 2003). However, there are some aspects also in the agent model of this study that are needed for facilitating the process automation specific requirements of control operations and which make this specification different to the references. Firstly, the BDI-model is applied to both discrete and continuous control operations. The BDI-model is proposed as the common programming model for MAS-based control applications in process automation. For this purpose the model was extended with the possibility for cyclic operation. The applications in the later chapters (chapters 4 and 5) will show how this model may be applied to both sequential and supervisory continuous control operations. Secondly, the qualitative process models are proposed to be used for identifying coordination needs between the agents. This design is needed for modelling the inter-relations between the process variables of continuous processes.

3.4.2 Effects on the properties of automation

The specification of the process automation agent platform affects the performance of the control applications through the mechanisms it offers for them. The mechanisms are required to enable both sequential and supervisory continuous control operations. The BDI-agent model with its extension for cyclic operation is designed for this requirement. The completeness, response time and synchronisation properties of the mechanisms of the platform also affect the performance of the platform. The completeness property concerns particularly the planning activity of agents and is dependent on the exact planning algorithm used. Also from the response time viewpoint the planning activity is possibly the problematic part of the platform, because for most planners it is not possible to give guaranteed response times. Another feature of the agent platform that affects the response time is the waiting time of the coordinator in the FIPA Contract Net protocol. Considering synchronisation of the control actions the important mechanisms of the platform are the execution

activity and the FIPA Request interaction protocol. The presented specification does not contain mechanisms for synchronisation of the control actions.

The specification of the agent platform for process automation is expected to enable enhancement of some properties of a process automation system. The property which is expected to be affected most directly is the reconfigurability of the automation system. The specification allows changes in the existence and capabilities of the agents as a means to adapt to configuration changes. The mechanisms of the platform for this purpose are similar to the earlier studies, i.e. the directory facilitator, the Contract Net based negotiation and the internal planning activity of the agents. There is a difference to the references, however; the agent society model adopted in this research allows reconfiguration among peers, which is expected to further facilitate the reconfigurability property. However, the usefulness of the approach for reconfigurability is dependent on the underlying automation system and the control applications implemented with the platform. The capability for reconfiguration is useful only if the automation system has some alternative configurations, e.g. through redundant equipment. The applications need to be designed for handling configuration changes in the scope of individual agents and coordinating the effects with other agents. Designs of such applications are studied in the following chapters (chapters 4 and 5). Enhanced system reconfigurability may also be assumed to reduce the complexity of automation systems engineering tasks, e.g. during maintenance operations.

The effect of the agent platform specification on the flexibility and responsiveness properties of process automation is less direct and is expected to be realised through the applications. However, process automation agents are proposed as a suitable means for designing applications for handling planned and unplanned operational changes and thus enhancing the flexibility and responsiveness properties. Designs of such applications are studied in the following chapters (chapters 4 and 5). The directory facilitator, negotiation and planning mechanisms of the platform are intended to be useful also in this context as already proposed in earlier studies. However, the remarks about the dependence on the characteristics of the underlying automation and adequately designed applications are valid also in this case. An important advantage of the platform in this research as compared to the references is that it allows usage of both discrete and continuous control operations when reacting to planned or unplanned events. Another advantage is that in some situations also operations related to flexibility and responsiveness can be performed among peer agents as allowed by the agent society model.

The specification of the process automation agent platform defines a model for designing applications. The principles of this model include agents as suitable system decomposition, goals as the major concept in an agent operation, planning as a suitable problem-solving method and negotiation as a suitable coordination mechanism. The applications are designed by configuring agents to the agent platform and programming applications in the form of plans of the agents. Although this kind of a MAS-based model of application design in automation has already been proposed in earlier studies (Tichy at al. 2002, Ingrand et al. 1992) there is not much experience on how this model affects the work of the application developers. However, an assumption that the MAS-based application design model might help reduce the complexity of automation systems has been presented (Jennings and Bussmann 2003). It should also be noted that all HMS-based applications in automation are not based

on MAS. An important different approach is to extend the function block programming model with mechanisms for agent-like cooperation (Neligwa and Fletcher 2003). The function block and MAS-based approaches can be regarded as alternatives.

3.4.3 Conclusions and open questions

The presented specification of the agent platform for process automation may be regarded as a prototype. However, it allows outlining some possible advantages and limitations of the approach. The most important expected benefit of the platform is the enhanced reconfigurability of the automation system, but this advantage is limited. The capability for reconfiguration is useful only to the extent that there is functional redundancy in the automation system, i.e. alternative ways to perform control operations. Another benefit is the possibility to utilise the mechanisms of the platform for enhanced flexibility and responsiveness through the applications implemented with the platform. The new application design model can be regarded as both an advantage and a limitation. On the one hand, the new model offers a new way to design distributed control applications. On the other hand, the application developers need to learn another quite different way of thinking about control applications. Another limitation of the platform is the response time of the planning activity, which is dependent on the planning task.

The presented form of the specification of the agent platform for process automation leaves some important open questions concerning its usage. One of the main open questions is how well the platform supports the design of useful applications. This question will be studied in the following chapters (chapters 4 and 5) with the description of two different experimental applications. Another open question concerns the extent of the effect of the agent-based approach on the properties of the automation system. Whereas this effect is assumed to be dependent on the applications, this question will be discussed again in conjunction with the applications. The third set of open questions concerns the possible ways to enhance and extend the specification of the agent platform. The known limitations of the platform could obviously be studied. In its present form the specification is also restricted because it is designed only for control functions. Other possible application functions for a MAS in process automation include e.g. monitoring. For monitoring applications other interaction protocols than the specified ones will be needed. Also a shared ontology of the agents would be needed to be extended for such applications. These questions have already been studied in other research efforts (Pirttioja et al. 2004). Other possible ways to extend the specification include e.g. inclusion of several directory facilitators, which has already been proposed in the references (Tichy at al. 2002).

4 Sequential control based on distributed planning

4.1 Introduction

This chapter presents the studies about distributed planning of sequential control actions. In these studies an application for sequential control (hereafter referred to as sequential control application) is developed using the agent platform for process automation described in Chapter 3. The application utilises the distributed planning methods discussed in Chapter 2.2.4. The target of the studies is an agent-based sequential control method that enables implementation of working sequential control functionality and can be argued to enhance some properties of process automation systems.

The studies described in this chapter consist of a specification of a distributed planning method for sequential control, experimentation with the method within a laboratory test environment and discussion of its properties. The specification contains models of the planning and plan execution methods both at agent society and agent levels. The specification defines how the agent platform for process automation can be used to build an application for sequential control. Experiments with a prototype application in the laboratory test environment are used for testing the feasibility of the method in simple scenarios. At the end of the chapter there is a discussion about the properties of the specified sequential control method and its relations to the previously published approaches.

4.2 Specification of the sequential control method

4.2.1 Planning at the agent society level

The society model of the sequential control application specifies how the process automation agents cooperate together in order to carry out sequential control actions on the target process. They follow the principles of the society model of the agent platform for process automation and utilise its cooperation mechanisms. However, the model of control goals and the cooperative problem-solving process formed by the actions of the agents is specific to this type of an application. An agent model complementing the agent society model is depicted in the following chapter (Chapter 4.2.2).

The sequential control application is a part of a higher-level automation layer that operates on top of an ordinary process automation system. It is one application running on the agent platform for process automation. The functional role of this application is to coordinate several control sequences of the lower level automation system. The lower level control sequences are preprogrammed control procedures whose purpose is to run the controlled process or a part of it into a particular end state. The lower-level control sequences are assumed to be distributed but inter-related. This means that they affect separate parts of the target process, but these separate control actions need to be coordinated in order to take the whole process into the desired goal state. The lower-level control sequences could also be executed on different units of hardware, e.g. process control stations. The type of operation expected from the sequential control application is deliberative. This means that the agents are required to verify the feasibility of the combination of distributed control sequences before the

execution of any part of the operation. Feasibility means that the combined sequence will not only end with the desired goal state, but will also satisfy other operational constraints attached to the sequential control operations.

The agent society model of the process automation agents is used as a means to organise the planning and execution of control sequences. According to the functional similarity of the agents each of them is obligated to plan and execute control sequences within its area of responsibility. Due to the semi-autonomy of the agents they can initiate planning and execution either because of their own goals, e.g. a goal invoked by monitoring activity, or because of negotiations with other agents. In the hierarchical agent organisation each agent is allowed to initiate a negotiation with its subordinates or its peers. In this way the supervisor agents can plan and execute control sequences indirectly via their subordinates and each agent is able to use the services of their peers as parts of their control sequences. The agents make queries to the directory facilitator of the agent society in order to identify those agents who can fulfil the needed goals.

The process automation agents utilise the concepts defined in the shared ontology of the agent society during the planning and execution of control sequences. The concepts of agent relations and services are used in order to identify suitable negotiation partners. Agent relations can be used to identify proper partners according to the rules of the agent society. Services can be used to identify agents that can fulfil the goals needed in the planning. During the planning the agents negotiate about the goals. Contracts are created at the final stage of planning and used during plan execution. The goals in control sequence planning describe the state that the controlled process should reach with a sequence. The goals may be represented with a name and a set of optional parameters. The parameters can be used to further specify the goal, e.g. with respect to goal status, the reference variables and their desired values (see Equations 4.1 and 4.2).

```
status_goal ::=  goal_name { parameter }                [4.1]
value_goal  ::=  goal_name variable value
                         { variable value }             [4.2]
```

Sequential control application follows the model of operation of the process automation agent society. The planning and execution of control sequences may be initiated either with an external request to some agent or through the monitoring activity among the agents. The actual planning and execution is then performed in two phases, both of which may be distributed and parallel processes among the agents. The planning process has some characteristics that are specific to this type of an application. The purpose of this process is to look for a shared plan for the agents that would take the controlled process to the requested goal state. The process can be characterised as a distributed planning process. It ends either with a shared plan for the requested control sequence or an indication of a failure to create one. After a successful planning process the created control sequence can be executed according to the general model of plan execution in the process automation agent society.

The planning of sequential control actions can be characterised as a distributed planning task (see Chapter 2.2.4). The actions of the possible control sequences and the precedence relations between them form a search space that the agents explore cooperatively. The agents are searching for an action plan that would take the target

process from the start state to the requested end state. In the following the distributed planning task is specified by describing the shared search space and the distributed search process.

The search space in the planning of sequential control actions is organised with the concepts of goals and plans as used in the agent platform for process automation. The search space is a directed graph, in which the nodes are goals and the arcs are subgoal relations. The search starts from the requested goal representing the end state of the whole sequence. During the search the goals are handled with plans. A plan is applicable to a goal if it can fulfil the goal and its preconditions are met. If the applicable plan contains subgoals new nodes and arcs are added to the search tree. If there are no subgoals in the plan the goal is marked as fulfilled. The search ends successfully when all goals are fulfilled, otherwise it fails. The search space contains alternative search paths if there is more than one plan applicable to any goal. The search space of the distributed planning of sequential control actions is illustrated in Figure 4.1.

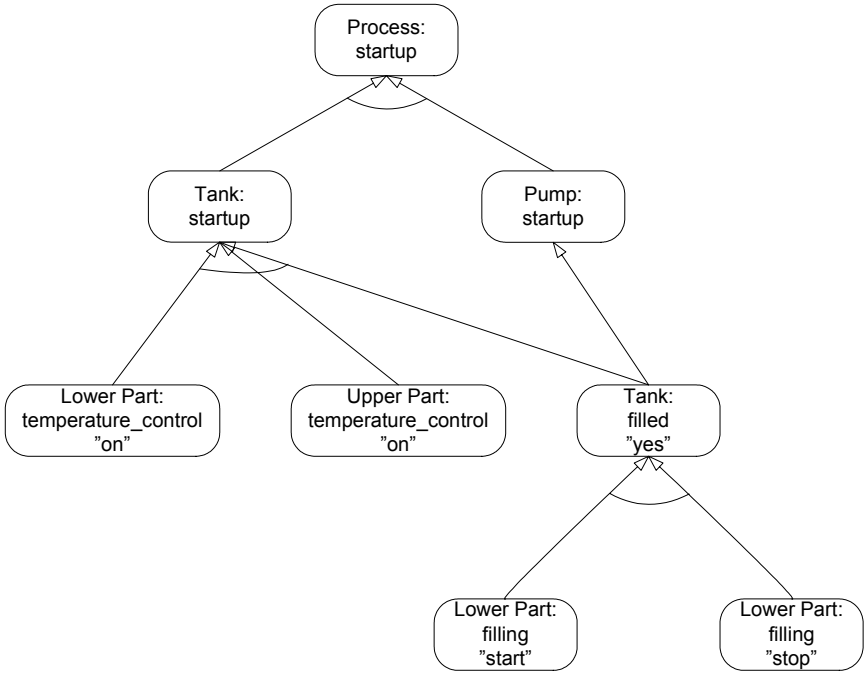


Figure 4.1 An example search space of the sequential control application represented as Tropos goal diagram.

The process automation agents perform the search for the shared control sequence following the rules of the process automation agent society and utilizing the internal planning and negotiation mechanisms of the individual agents. The search space is decomposed to different agents according to their capabilities. Each goal node is handled by an agent that has a plan applicable to the goal. The agent applies its own plans to the created subgoals, if possible, or it passes them to another agent via negotiation. In this way the internal search processes of agents are tied together into a cooperative search. The hierarchical organisation of the process automation agent society is used to organise the search process. The role of supervisor agents is to decompose the goals relating to their areas of responsibility into subgoals concerning smaller parts of the target process. They pass these subgoals to the subordinate agents supervising the respective process areas. Peer agents may pass goals to their peers in

order to get the preconditions of their plans fulfilled. The cooperative search process of the process automation agents for the shared control sequence is illustrated in Figure 4.2.

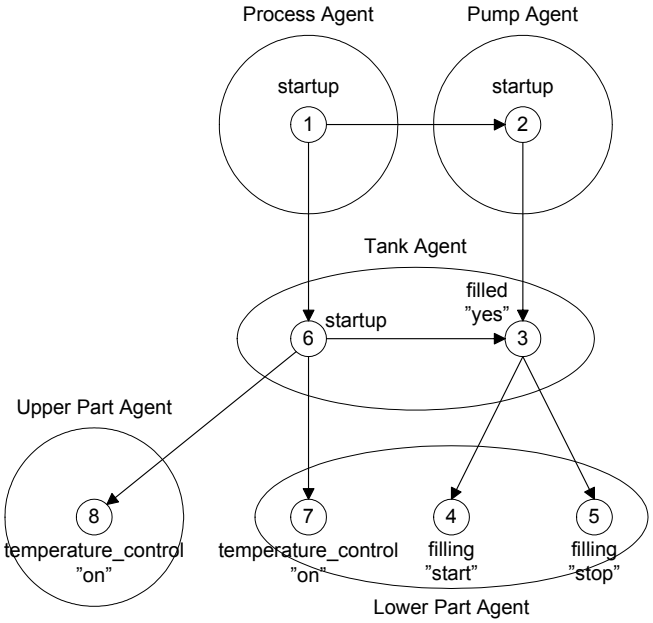


Figure 4.2 An example search process of the sequential control application. The numbers indicate the order of exploring the nodes in the search space.

The execution of the shared control sequence is based on the plans and contracts created in the planning phase. It is performed according to the plan execution model of the agent platform for process automation (see Chapter 3.2.1). Thus, plan execution is not specific to this kind of an application.

4.2.2 Planning at the agent level

The agent model of the sequential control application specifies how the process automation agents use their internal mechanisms in order to perform this particular application. They utilise the generic planning, plan execution and process modelling capabilities of a process automation agent (see Chapter 3.2). The application-specific part of agent behaviour is specified with plans in the agent plan library. These plans are used to implement the decision-making needed in each agent during the cooperative planning process specific to this type of an application.

The purpose of plans in the sequential control application is to encapsulate control sequences so that they can be combined during the planning process. According to the plan representation used in the plan library of a process automation agent the plans are a combination of a goal, a context and a body (see Chapter 3.2.2). In this application the goal is used to represent the state of the controlled process to which this sequence will lead when executed. The context describes the logical conditions of the state in which the sequence can be used. The body contains the definition of the actual sequence that implements the state change to the goal state from the initial state with the given conditions. The plan body is represented as a procedure containing control actions, local and negotiated subgoals and control logic combining these. Local subgoals are goals for other sequences within the same agent. Negotiated subgoals

identify state changes that are required for the sequence but are not controllable by the agent owning it. An agent can get these subgoals fulfilled via negotiations with other agents. The plans of all agents follow this same model regardless of their role in a search. A skeleton of a plan in the sequential control application is presented in Figure 4.3.

```
// Pseudocode of a plan for the sequential control application
Plan: {
  GOAL: ACHIEVE control_goal_name $param;
  CONTEXT:
    // Parameters are meanigful
    (== $param "value");
    // Situation is suitable
    FACT var $var;
    (== $var value);
  BODY:
    // Local control action
    EXECUTE PlanControlAction "var" "value";
    // Local subgoal
    ACHIEVE local_subgoal;
    // Negotiated control goal
    EXECUTE Negotiate "negotiated_subgoal" "param_x";
}
```

Figure 4.3 Pseudocode of an example plan in the sequential control application represented with the plan language of JAM.

The internal planning activity of a process automation agent in the sequential control application utilises the generic planning capabilities of the agent in order to fulfil its role in the distributed search process of the society level. The responsibility of a single process automation agent in this search process is to perform the search within its area of responsibility and connect it to the search processes of other agents when needed. For this purpose the agent has to manage its run-time data structures associated to the planning processes, perform its local planning task and take part in the necessary negotiations with other agents. Although these activities are mostly implemented by the agent platform for process automation (see Chapter 3), they are used in a way particular to this application.

In the sequential control application the run-time data structures of a process automation agent are needed particularly for storing the state of the distributed planning process. The state is stored in a distributed way in all agents that are involved in the planning process. The run-time data includes goals, plans and contracts of the agents that form a shared plan of the agents. Due to the deliberative nature of this application the run-time data needs to be stored until the entire planning process of the agent society level has ended and the plans are executed (see Chapter 4.2.1).

In the sequential control application the local planning process of a process automation agent is used in a deliberative way as part of the distributed planning process. The local planning processes of all the agents are similar. The input to the local planning process of an agent consists of the requested goal, other goals of the agent and the current measurement values. The outcome from the planning activity is an updated set of plans and contracts.

The purpose of the negotiation activity in the sequential control application is to extend the distributed search space from one agent to another one. The negotiation is performed according to the Contract Net protocol implemented in the agent platform. The decision-making required at each step of the negotiation protocol and the exact content of the messages exchanged during the negotiation are specific to this type of an application. The decision-making and messages are characterised by their role as a part of a distributed planning process. The negotiation activity as it appears in this application is illustrated in Figure 4.4 and explained in more detail below.

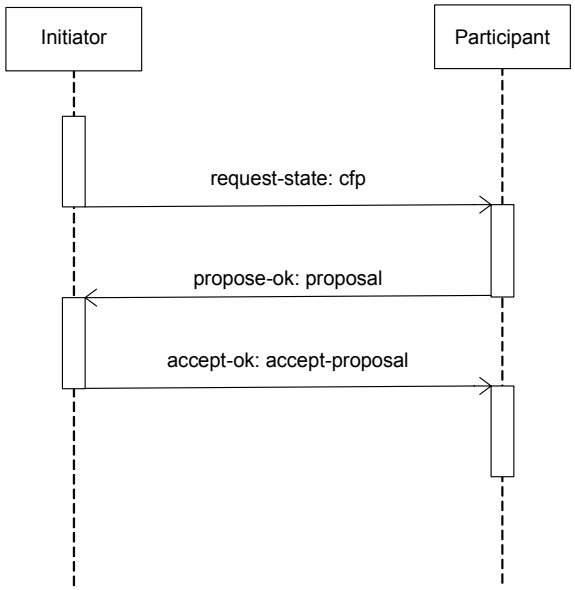


Figure 4.4 Negotiation between two process automation agents in the sequential control application represented as an AUML sequence diagram.

At the CFP creation stage, the decision making task of the initiator agent is to select partners for negotiation. The partners are identified according to their services and organisational role. The initiator selects those peer or subordinate agents that provide a service with the needed goal. The CFP message contains only the description of the goal, i.e. its name and optional parameters. At this stage the agents can mark the potential contract as called-for.

At the proposal creation stage, the decision making task of the participant agent is to determine if it can fulfil the goal specified in the CFP message. The participant performs this by initiating a local planning task for the goal in the CFP and other goals it has. If the planning task ends successfully a proposal message is sent to the initiator. The proposal message may contain only a positive answer to the CFP or it may also contain additional information concerning the proposal, e.g. cost of the proposed contract. At this stage the agents can mark the potential contract as proposed.

At the answer creation stage, the decision-making task of the initiator is to select one of the proposals. It can select the proposal with minimum cost if this information is included or based on some application-specific decision rule implemented in a plan. If there are no proposals a failure to fulfil the goal is passed to the local planning activity. An accept-proposal message is sent to the author of the chosen proposal and reject-proposal messages to other participants. At this stage the agents can mark the contract as confirmed.

The participant does not have any real decision making to do at the answer handling state. It just needs to pass the acceptance further to other agents contracting subgoals to this one.

4.3 Experiments with the sequential control method

4.3.1 Specification of the experiments

The laboratory test environment with its experimental implementation of the process automation agent platform has been used to test the sequential control application. Two different application sequences, process startup and shutdown, were designed, implemented and tested. The purpose of these experiments was to demonstrate the operation of the application and test its feasibility in simple scenarios. Due to the simplicity and small number of the test scenarios it is not possible to make any thorough assessment of the properties of the control method based on these experiments. The properties of the method are discussed more in Chapter 4.4.

The objective of the process startup scenario is to run the test process (see Chapter 3.3) from a shutdown state into a normal operation state. In the shutdown state the control loops of the automation application are not in operation, the water circulation is off and the tank is not necessarily full. The normal operation state is opposite to the shutdown state. The tank is full of water, the water level control loop and the two water temperature control loops at different parts of the tank are operational and the water circulation is on. The state change is carried out with execution of a partially ordered sequence of control actions (see Figure 4.5). Some control actions need to be executed in sequence whereas some others may be executed in parallel. The control actions are those available for the devices in the automation system. Also the pre-conditions of the control actions have to be taken into account in the control sequence. In this scenario the pre-conditions refer to the status of devices, i.e. they have to be in usable condition, and to the status of the process, e.g. a full tank does not need to be filled.

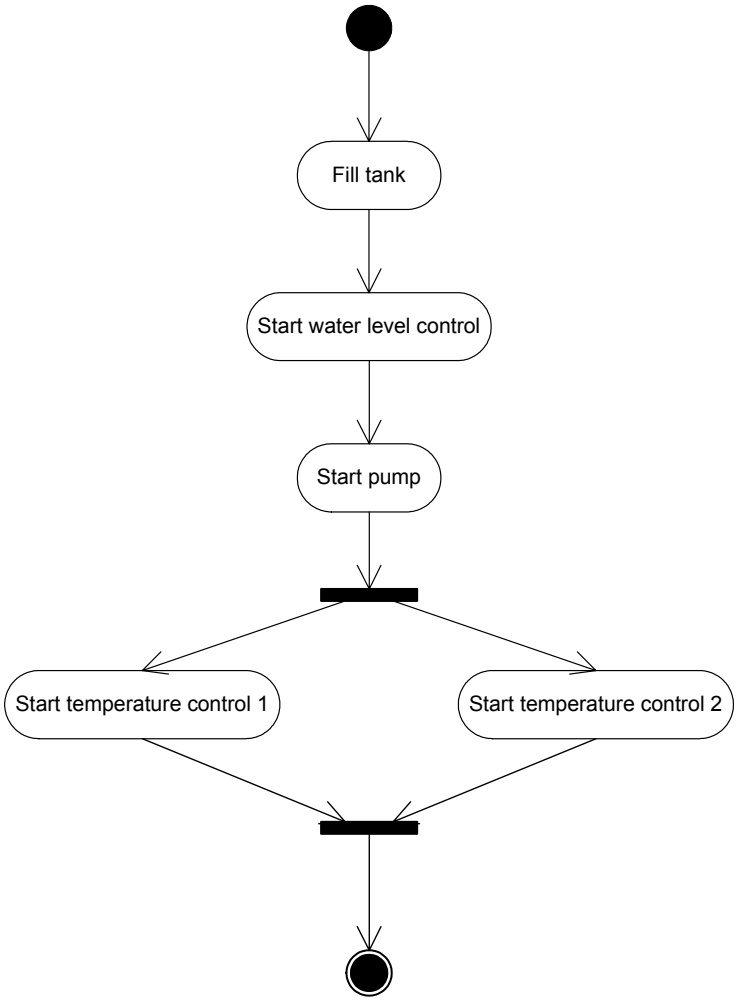


Figure 4.5 Startup sequence of the test process represented as an UML activity diagram.

The process shutdown scenario is opposite to the process startup scenario. The purpose of this scenario is to run the process from the normal operation state to the shutdown state. This scenario is maybe somewhat simpler than the startup scenario. It is included in the experiments as an additional example (see Figure 4.6).

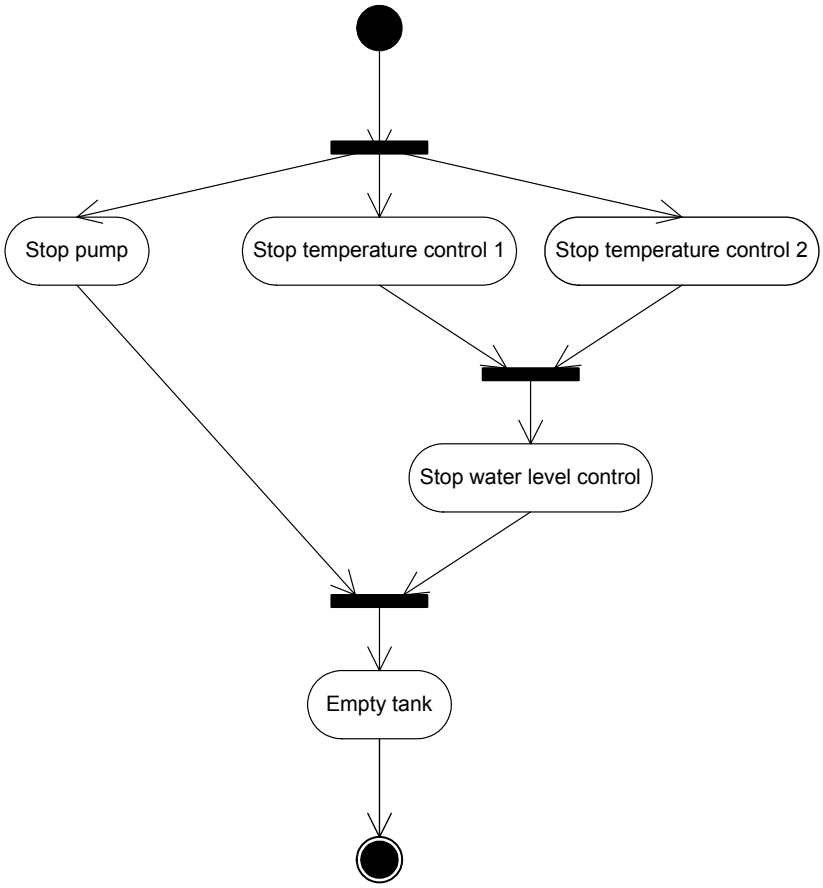


Figure 4.6 Shutdown sequence of the test process represented as an UML activity diagram.

4.3.2 Application design for the experiments

The applications for the test process startup and shutdown scenarios were defined as plans that were configured into the plan libraries of the agents in the test environment. For the experiments, the plans needed for the startup and shutdown sequences were defined for all five agents of the test agent application (see Chapter 3.3.2).

The Process Agent has a special role in the startup and shutdown applications as the topmost agent of the agent society. It provides the process startup and process shutdown goals as services that can be used to initiate planning and execution of these sequences. The plans for fulfilling these goals decompose the top-level goals into subgoals for subordinate agents. The startup and shutdown plans of the Process Agent are illustrated in Figure 4.7. These plans are rather similar and simple. Their purpose is to decompose the top level goal into subgoals and pass them to the subordinate agents.

```
// Startup sequence of the whole process
Plan: {
  GOAL: ACHIEVE startup;
  BODY:
    // Request startup of subprocesses
    EXECUTE Negotiate "startup" "pump" "non-blocking";
    EXECUTE Negotiate "startup" "tank";
}

// Shutdown sequence of the whole process
Plan: {
  GOAL: ACHIEVE shutdown;
  BODY:
    // Request shutdown of subprocesses
    EXECUTE Negotiate "shutdown" "pump" "non-blocking";
    EXECUTE Negotiate "shutdown" "tank";
}
```

Figure 4.7 Pseudocode of the startup and shutdown plans of the Process Agent represented with the plan language of JAM.

The Pump Agent and the Tank Agent are subordinates of the Process Agent. They also provide startup and shutdown goals as services. In addition to this, they have other goals as services, e.g. fill tank by Tank Agent. Other agents can negotiate about these goals when planning their control sequences. The startup and tank filling plans of the Tank Agent are illustrated in Figure 4.8. The startup plan utilises the planning method in a versatile way. It contains a local action, a local subgoal and two negotiated goals. Tank filling is designed as a separate plan, so that it can be provided also as a separate service and be used also in other situations than startup.

```
// Startup sequence of the tank subprocess
Plan: {
  GOAL: ACHIEVE startup;
  BODY:
    // Local subgoal for filling the tank
    ACHIEVE filled "yes";
    // Start water level control
    EXECUTE PlanControlAction "evsp" "1650";
    // Request starting of temperature controls
    EXECUTE Negotiate "temperature_control on" "upper_part" "non-
bloking";
    EXECUTE Negotiate "temperature_control on" "lower_part";
}

// Sequence for filling the tank
Plan: {
  GOAL: ACHIEVE filled $status;
  CONTEXT:
    // Goal is to fill the tank
    (== $status "yes");
    // Tank is not filled yet
    FACT lm $lm;
    (< $lm 1600);
  BODY:
    // Request start of filling
    // Keep filling until tank filled and stop
    EXECUTE Negotiate "filling" "start" "blocking";
    EXECUTE PlanConditionalAction "lm" "1600" "greater";
    EXECUTE Negotiate "filling" "stop";
}
```

Figure 4.8 Pseudocode of the startup and tank filling plans of the Tank Agent represented with the plan language of JAM.

The Upper Part Agent and the Lower Part Agent are subordinates of the Tank Agent. They provide services for starting and stopping their control loops. The Lower Part Agent also has services for filling the tank. The plans of the Upper Part Agent for starting its water temperature control loop are illustrated in Figure 4.9. This kind of a plan is typical for the lowest level agents in the hierarchical agent society. They contain only local control actions.

```
// Sequence for puting the water temperature control on
// at the upper part of the tank"
Plan: {
  GOAL: ACHIEVE temperature_control $status;
  CONTEXT:
    // Goal is to start the control
    (== $status "on");
    // Control is not on yet
    FACT cvm11 8;
    FACT cvm12 8;
  BODY:
    // Check that equipment is ok
    RETRIEVE cvs11 $status11;
    RETRIEVE cvs12 $status12;
    WHEN : TEST (|| (== $status11 1) (== $status12 1)) { FAIL; };
    // Plan the actual sequence
    EXECUTE PlanControlAction "mvsp11" "0";
    EXECUTE PlanControlAction "mvsp12" "0";
    EXECUTE PlanControlAction "tsp1" "31.0";
    EXECUTE PlanControlAction "cvm11" "12";
    EXECUTE PlanControlAction "cvm12" "12";
}
```

Figure 4.9 Pseudocode of the plan needed for starting the water temperature control loop of the Upper Part Agent represented with the plan language of JAM.

4.3.3 Results from the experiments

The sequential control application was tested with three different variations of the process startup scenario and one variation of the process shutdown scenario. The variations demonstrate the basic capabilities of the cooperative planning process to adapt to different states of the controlled process and the status of the control devices.

The first variation of the process startup scenario is a basic one. The start state of the process is a proper shutdown state and all the equipment is usable. The cooperation among the agents during the distributed planning process of the startup sequence is illustrated in Table 4.1. The planning process proceeds downwards in the agent society. There is also one peer-to-peer negotiation between the Pump Agent and the Tank Agent. The cooperation during the execution of the startup sequence is illustrated in Table 4.2. The process follows the contracts created during the planning stage. The functions of the control system are started in the proper order.

Table 4.1 Message exchange among the agents during the planning of the process startup sequence in the first variation of the test scenario.

No	Sender	Receiver	Message type	Message content
1	Process	Pump	cfp	goal: startup
2	Pump	Tank	cfp	goal: filled
3	Tank	Lower Part	cfp	goal: filling start
4	Lower Part	Tank	proposal	
5	Tank	Lower Part	cfp	goal: filling stop
6	Lower Part	Tank	proposal	
7	Tank	Pump	proposal	
8	Pump	Process	proposal	
9	Process	Tank	cfp	goal: startup
10	Tank	Upper Part	cfp	goal: temperature_control on
11	Upper Part	Tank	proposal	
12	Tank	Lower Part	cfp	goal: temperature_control on
13	Lower Part	Tank	proposal	
14	Tank	Process	proposal	
15	Process	Pump	accept_proposal	
16	Process	Tank	accept_proposal	
17	Pump	Tank	accept_proposal	
18	Tank	Lower Part	accept_proposal	
19	Tank	Lower Part	accept_proposal	
20	Tank	Upper Part	accept_proposal	
21	Tank	Lower Part	accept_proposal	

Table 4.2 Message exchange among the agents during the execution of the process startup sequence in the first variation of the test scenario.

No	Sender	Receiver	Message type	Message content
1	Process	Pump	request	goal: startup
2	Pump	Tank	request	goal: filled
3	Tank	Lower Part	request	goal: filling start
4	Process	Tank	request	goal: startup
5	Lower Part	Tank	inform	
6	Tank	Lower Part	request	goal: filling stop
7	Lower Part	Tank	inform	
8	Tank	Pump	inform	
9	Tank	Upper Part	request	goal: temperature_control on
10	Tank	Lower Part	request	goal: temperature_control on
11	Pump	Process	inform	
12	Upper Part	Tank	inform	
13	Lower Part	Tank	inform	
14	Tank	Process	inform	

The second variation of the process startup scenario is similar to the basic scenario, but the process start state is different. In this variation the tank is already full in the beginning of the scenario. The cooperation among the agents during both planning and plan execution processes is quite similar to the basic variation. Only the negotiation between the Tank Agent and the Upper Level Agent about tank filling is unnecessary as illustrated in Table 4.3. This change is not visible to other agents, e.g. the Pump Agent does not need to notice it.

Table 4.3 Message exchange during the planning process of the startup sequence in the second variation of the test scenario. In this variation the tank is already full in the beginning of the scenario.

No	Sender	Receiver	Message type	Message content
1	Process	Pump	cfp	goal: startup
2	Pump	Tank	cfp	goal: filled
7	Tank	Pump	proposal	
8	Pump	Process	proposal	
9	Process	Tank	cfp	goal: startup
10	Tank	Upper Part	cfp	goal: temperature_control on
11	Upper Part	Tank	proposal	
12	Tank	Lower Part	cfp	goal: temperature_control on
13	Lower Part	Tank	proposal	
14	Tank	Process	proposal	
15	Process	Pump	accept_proposal	
16	Process	Tank	accept_proposal	
17	Pump	Tank	accept_proposal	
20	Tank	Upper Part	accept_proposal	
21	Tank	Lower Part	accept_proposal	

The third variation of the process startup scenario is similar to the basic scenario, but the equipment status is different. In this variation one of the valves is not operational. It is not possible to create a feasible startup sequence, because one of the irreplaceable resources is not operational. This situation is detected during the planning process among the agents that end with an indication of the unfeasibility of the planning task. For example, if one of the valves needed for filling the tank is faulted then the planning process will stop after Message number 3 in Table 4.1.

The process shut down scenario was tested with only one variation. The cooperation among the agents during the planning process of the shutdown sequence is illustrated in Table 4.4 and the cooperation during the execution of the sequence in Table 4.5. The planning process proceeds downwards in the agent society. There is no need for peer-to-peer negotiation in this scenario.

Table 4.4 Message exchange among the agents during the planning of the process shutdown sequence.

No	Sender	Receiver	Message type	Message content
1	Process	Pump	cfp	goal: shutdown
2	Pump	Process	proposal	
3	Process	Tank	cfp	goal: shutdown
4	Tank	Upper Part	cfp	goal: temperature_control off
5	Upper Part	Tank	proposal	
6	Tank	Lower Part	cfp	goal: temperature_control off
7	Lower Part	Tank	proposal	
8	Tank	Process	proposal	
9	Process	Pump	accept_proposal	
10	Process	Tank	accept_proposal	
11	Tank	Upper Part	accept_proposal	
12	Tank	Lower Part	accept_proposal	

Table 4.5 Message exchange among the agents during the execution of the process shutdown sequence.

No	Sender	Receiver	Message type	Message content
1	Process	Pump	request	goal: shutdown
2	Process	Tank	request	goal: shutdown
3	Tank	Upper Part	request	goal: temperature_control off
4	Pump	Process	inform	
5	Tank	Lower Part	request	goal: temperature_control off
6	Upper Part	Tank	inform	
7	Lower Part	Tank	inform	
8	Tank	Process	inform	

4.4 Discussion

The specification of the sequential control application is assessed in here by comparing it to related sequence planning approaches and discussing its properties. Similarities and differences to the references are characterised and justifications to the adopted design choices are presented. The effect of these choices to the automation system properties is discussed and compared to the references. Finally, conclusions are drawn based on this study and open questions concerning the research topic are outlined.

4.4.1 Design choices in the specification

The society model of the sequential control application shares many features with previously reported research (see Chapter 2.3). Maybe the most noteworthy reference to this application is the CWS study (Maturana at al. 2002, 2003, Tichy at al. 2002). The similarities include deliberative operation, spatial decomposition of the search space, usage of a directory facilitator and negotiation based on Contract Net. However, the specified agent society model also has some features that are different from the references. One difference is the utilisation of both the vertical and horizontal cooperation channels of the agent platform. This design separates goal-decomposing, vertical coordination from synchronizing, horizontal coordination. In this way a supervisor agent does not need to plan the order in which its subordinates to perform actions for its goals. Another difference is the central role of goals in the shared ontology. In the presented specification the concept of goals is used as a combining element both in the internal planning of the agents and the coordination between them. According to this design the internal planning of an agent and the coordination in the agent society are both parts of a goal-oriented search processes.

The agent model of the sequential control application is quite similar to some previously reported research (see Chapter 2.3). Important related research efforts in this context include CWS (Tichy at al. 2002, Maturana at al. 2002, 2003), PRS (Ingrand at al. 1992), manAge (Heikkilä at al. 1999) and CASA (Flake at al. 2001). The similarities include BDI or PEM as agent model, planning as a problem solving method and usage of procedural plans. The most important difference of the specified agent model to the references is that it is designed to co-exist in the same agents with another type of an application for continuous control (see Chapter 5), which is a particular requirement in process automation.

4.4.2 Effects on the properties of automation

The performance of the sequential control application depends on both its properties as a distributed planning method and its properties as a control application. The distributed planning method needs to solve the coordination problem of distributed planning (see Chapter 2.2.4) in such a way that it fulfils the requirements of the control application. The important properties of the method in this sense are completeness and time complexity of the distributed search algorithm. An important aspect of the algorithm that affects these properties is how many agents are allowed to be planning at the same time and whether the possible parallel planning activities are synchronised. In parallel, asynchronous planning the agents could e.g. initiate several negotiations without waiting for the results from the previous ones. This kind of planning is quite complicated and would require more advanced coordination mechanisms than defined in the specification. However, it could result in better scalability. The startup and shutdown test applications define a simple search strategy as a demonstration. In these experimental applications the search strategy is best-first search, in which only one agent is allowed to plan at any one time.

The specification of the sequential control application is expected to facilitate the reconfigurability property of process automation systems. The effect on the reconfigurability property is assumed to originate from the combination of the mechanisms of the agent platform and the applications. The sequential control application is based on the distributed planning process that utilises the directory

facilitator, the Contract Net based negotiation and the internal planning activity provided by the platform. With the combination of these mechanisms it is possible to re-plan sequential control operations in the case of changes in the existence and capabilities of the agents. These changes may reflect modifications of the underlying automation system, e.g. device replacement and maintenance operations. The presented mechanism works if the reconfiguration can be done through re-planning the combination of the sequences wrapped in the plans. Even though the design of this application is slightly different from the CWS the expected effect on the reconfigurability property is quite analogous. The main difference is the possibility to include continuous controllers in the reconfiguration process through other applications of the platform (see Chapter 5).

The specification of the sequential control application is also expected to enable enhancement of the flexibility and responsiveness properties of process automation systems. The intended means for this is to design sequential control applications that can handle planned or unplanned operational change situations and conform to the presented model of distributed planning. The distributed planning process is able to create new sequences as a combination of the sequences wrapped in the plans. If operational change situations can be handled with these new sequences then the specification can facilitate flexibility and responsiveness. The situations intended to be handled in this way are process state changes, e.g. in batch processes, product or raw material changes and device malfunctions. Plans for handling situations like these are needed in order to the approach to be feasible. Also in this case this application is similar to the CWS.

The specification of the sequential control application shows how the application design model of the agent platform can be used for developing sequential control applications. The applications are designed by wrapping sequences of lower-level automation system inside plans and attaching information about their end state and preconditions to the plans. This kind of an approach for designing sequential control applications in automation is quite new, although similar approaches have already been proposed in earlier studies, e.g. CWS and PRS. This model of application development is dependent on a few assumptions. It is assumed that goal and precondition information is enough to determine the situations when a plan and its sequence are applicable. It is also expected that the application developers are able to trust the distributed planning process and are able to verify their applications in a sufficient manner. Possible errors in plans, e.g. logically contradicting preconditions, lead to a failure to reach the given goal during a planning process.

4.4.3 Conclusions and open questions

The specification of the sequential control application may be characterised as an aggregate design and the experimental startup and shutdown applications as demonstrating prototypes. However, it is possible to recognise the basic advantages and limitations of the application. The main advantage of the application is the enabled positive effect on the adaptation properties of automation. Another advantage is that the application can be implemented using a platform which is designed to be able to run also continuous control functions of process automation. The main limitation of the application concerns the distributed planning process used in it. There is not enough knowledge about the performance of different distributed search strategies that could be used in this application. Also the characteristics of the

underlying automation system and the controlled process affect the usefulness of the sequential control application. Particularly the possible lack of redundancy in them, i.e. alternative sequences to reach a goal state, restricts the adaptation properties. The meaning of the experiments described in this study is that they are able to demonstrate the characteristics of the distributed planning process and the design of plans for it in simple test scenarios.

The presented form of the specification leaves important open questions concerning the utilisation of MAS in sequential control applications in process automation. Maybe the most important open questions are if the presented aggregate specification of distributed planning is feasible in a more general case and how to make it more detailed so it could be better evaluated. These questions could be answered by studying the possible distributed search strategies more closely. More advanced search strategies could be studied with this application, e.g. asynchronous and parallel search. The important properties of the search strategies to be studied are completeness and time complexity. Another property to test is the scalability of the search with respect to the number of agents. Test scenarios with larger search spaces and larger numbers of agents could be useful, e.g. in the form of simulations. Finally, interleaving of planning with plan execution would also need to be developed.

5 Supervisory control based on distributed search

5.1 Introduction

This chapter presents the studies about supervisory control based on distributed search. In these studies an application for supervisory continuous control (hereafter referred to as supervisory control application) is developed using the agent platform for process automation described in Chapter 3. The application utilises the distributed search methods discussed in Chapter 2.2.3. The target of the studies is an agent-based supervisory control method that enables implementation of working supervisory control functionality and can be argued to enhance some properties of process automation systems.

The studies described in this chapter consist of a specification of a distributed search method for supervisory control, experimentation with the method within a laboratory test environment and discussion of its properties. The specification contains models of the search method both at agent society and agent levels. The specification defines how the agent platform for process automation can be used to build an application for supervisory control. Experiments with a prototype application in the laboratory test environment are used for testing the feasibility of the method in simple scenarios. At the end of the chapter there is a discussion about the properties of the specified supervisory control method and its relations to previously published approaches.

5.2 Specification of the supervisory control method

5.2.1 Search at the agent society level

The society model of the supervisory control application specifies how the process automation agents cooperate in order to carry out supervisory control actions on the target process. They follow the principles of the society model of the agent platform for process automation and utilise its planning and negotiation mechanisms. However, the model of control goals and the cooperative problem-solving process formed by the actions of the agents is specific to this type of an application. An agent model complementing the agent society model is depicted in the following chapter (Chapter 5.2.2).

The supervisory control application is a part of a higher-level automation layer that operates on top of an ordinary process automation system. It is one application running on the agent platform for process automation. The functional role of this application is to coordinate the operation of several continuous controllers of the lower level automation system. These lower level controllers are regulatory controllers whose purpose is to keep their control variables at their set-points. The lower level controllers are assumed to be distributed but inter-related. This means that they control separate parts of the target process, but their control operations need to be coordinated because they affect each other through the dynamics of the target process. The lower-level controllers may also be running on different units of hardware, e.g. process control stations or intelligent instruments. The type of operation expected from the supervisory control application is reactive and cyclic. This means that the agents only plan a limited set of control actions and then execute them immediately.

The effect of the control actions is evaluated via feedback from the target process. This information is then utilised at the following control cycles.

The agent society model of the process automation agents is used as a means to organise the supervisory control application. According to the functional similarity of the agents each of them is obligated to supervise the lower level controllers within its area of responsibility. Due to the semi-autonomy property of the agents they can initiate supervisory control actions either because of their own goals, e.g. a goal invoked by monitoring activity, or because of negotiations with other agents. In the hierarchical agent organisation each agent is allowed to initiate negotiation with its subordinates or peers. Supervisor agents can coordinate their subordinates by initiating negotiations or they can let their subordinates do the coordination among themselves and just set limits to their operation. In the latter case one of the subordinates becomes a coordinator. However, this role is not predefined among the subordinates and depends on the situation and capabilities of the agents. The agents make queries to the directory facilitator of the agent society in order to identify those agents who can affect the needed control variables.

The process automation agents utilise the concepts defined in the shared ontology of the agent society during the search of supervisory control actions. The concepts of agent relations and services are used in order to identify suitable negotiation partners together with the qualitative process model. The process model can be used to make inferences about which other control variables can affect a particular control variable. Services can then be used to identify agents that control these control variables. Agent relations can be used to check the acceptability of partner agents according to the rules of the agent society. The negotiations between agents in this application concern about goals that describe changes in the process variables due to supervisory control actions. The goals may be represented with a name, a process variable name and the desired value of change as illustrated in Equation 5.1. In this application the concept of contract is not necessary, because contracted actions are performed immediately after negotiation.

```
change_goal ::=  goal_name variable change
                          { variable change }          [5.1]
```

The supervisory control application follows the general model of operation in process automation agent society. The application may be initiated either with an external request to some agent or via the monitoring activity among the agents. The actual search is then performed in distributed and iterative fashion. The search process has some characteristics that are specific to this type of an application. The purpose of this process is to enhance the control of the process by looking for better values for a set of supervisory control variables distributed among some of the agents. The process can be characterised as distributed and iterative optimisation. It ends either with new values for the set of distributed control variables or an indication of a failure to control the process acceptably. During iterations decision-making through negotiation and control action execution is interleaved. After each iteration step the decided control actions are executed before the next one.

The supervisory control activity can be characterised as an iterative search task (see Chapter 2.2.3). The possible values of supervisory control variables and their constraints form a search space that the agents explore iteratively. The agents are

searching for values of control variables that would optimise a given objective function, e.g. a square sum of control errors, or at least produce a good enough result. The objective function is a function of some of the measured variables of the target process. These variables are affected directly or indirectly by the control variables. In the following the iterative search task is specified by describing the search process as a whole and the operation at each iteration step.

The search process in the supervisory control activity is organised as iterative refinement of the values of a set of control variables. A group of process automation agents are iteratively changing a set of inter-related supervisory control variables in order to find better values for them in a particular situation. The group is formed at the first iteration step via negotiation initiated by one of the agents. When an agent observes a need for changing its control variables it can use its qualitative process model in order to find other control variables affecting its control objective. After this it can identify agents supervising these variables via the directory facilitator. This agent then becomes the coordinator of the distributed search process. The task of the coordinator is to make sure that the actions of all agents in the group, including itself, form a meaningful search. It does this via bilateral negotiations with each of the other agents in the group. The search process of the supervisory control application is illustrated in Figure 5.2.

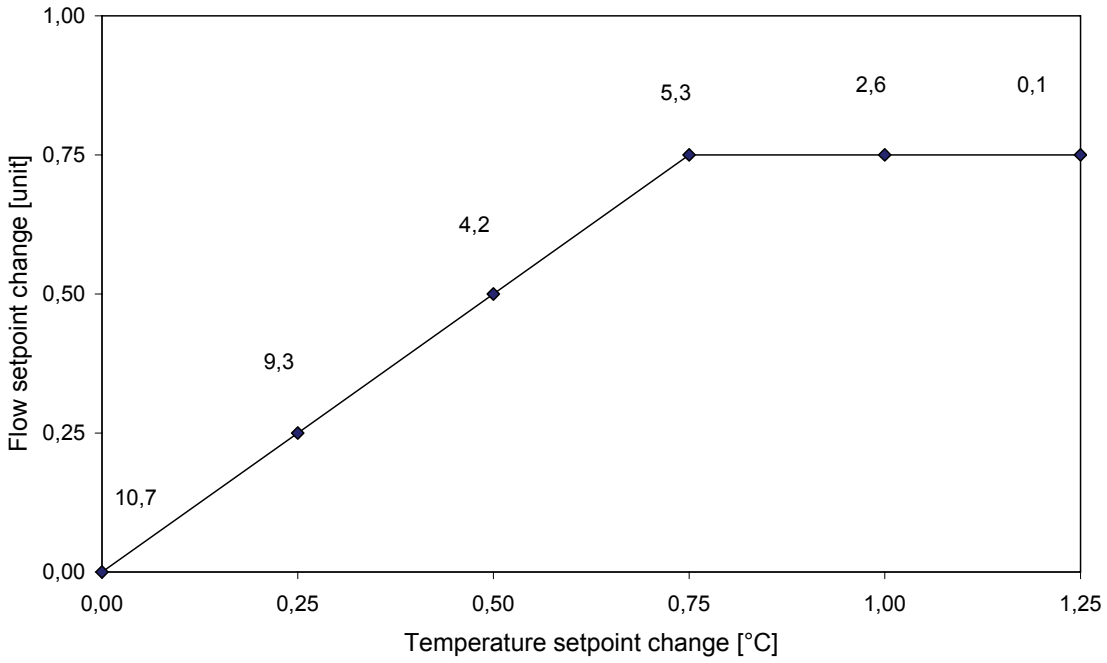


Figure 5.1 An example search space of the supervisory control application. The numbers inside the figure indicate the values of the objective function at each iteration step.

At each iteration step decision-making takes place via negotiation between a set of process automation agents. The agents perform the decision-making required in the negotiation based on their plans. At each negotiation step the agents can measure feedback from the controlled process and utilise this information in their decision-making (see Figure 5.2). At some stage the coordinator agent is expected to terminate the search process. The negotiation protocol of the agents is the FIPA Contract Net

Protocol. The agent coordinating the search has the role of initiator while other agents are participants. The messages exchanged during the negotiation have a somewhat different meaning than in the original Contract Net. The exact form of negotiation among the agents including the form of messages and internal decision procedures of the agents is described in more detail in Chapter 5.2.2.

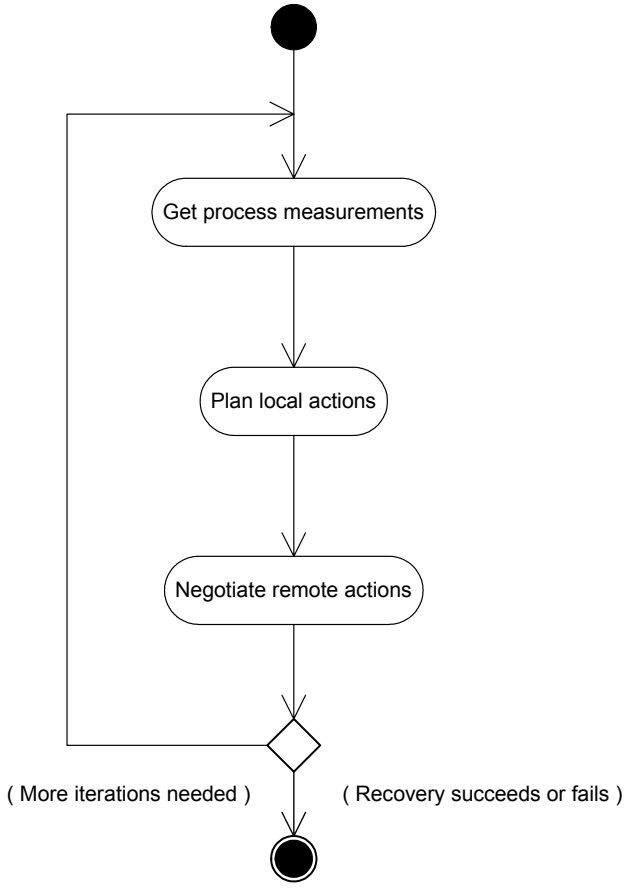


Figure 5.2 Iterative search process of the supervisory control application represented as an UML activity diagram.

The execution of the shared control actions is based on the plans created during the negotiation. They are executed immediately after the negotiation at each iterative search step. The execution is performed according to the plan execution model of the agent platform for process automation (see Chapter 3.2.1). Thus, plan execution is not specific to this kind of an application.

5.2.2 Search at the agent level

The agent model of supervisory control application specifies how the process automation agents use their internal mechanisms in order to perform this particular application. They utilise the generic planning, plan execution and process modelling capabilities of a process automation agent (see Chapter 3). The application-specific part of agent behaviour is specified with plans in the agent plan library. These plans are used to implement the decision-making needed in each agent during the iterative search process specific to this type of an application.

The purpose of plans in the supervisory control application is to encapsulate decision logic for setting the values of control variables. The generic plan representation of the

plan library of a process automation agent is used, i.e. the plans are a combination of a goal, a context and a body. In this application the goal is used to represent an intended action of an agent in a cooperative supervisory control operation, i.e. an intention to change the values of some control variables. The context describes the logical conditions for the negotiation situation and the state of the target process when this plan is feasible. The body contains the decision logic of how to change the values of the control variables. There are two different types of plans: those referring to local changes and those referring to negotiated changes. The plans representing local changes are similar to the plans in the sequential control application. They encapsulate a sequence of supervisory control actions and associate it with a local subgoal. The plans representing the negotiated changes define the decision logic of an agent during a negotiation. They are tied to the role of the agent in the negotiation, i.e. an initiator or a participant. The former has a plan to make a call for proposal and a decision in a negotiation. The latter has a plan to make a proposal. Skeletons of plans in the supervisory control application are presented in Figure 5.4 and Figure 5.4.

```
// Pseudocode of a plan for an initiator in the supervisory
// control application
Plan: {
  GOAL: ACHIEVE situation_goal $var $param;
  CONTEXT:
    // This plan applies only for control of a particular variable
    (== $var "var");
  BODY:
    // Retrieve decision variables
    RETRIEVE iter_num $iter_num;
    RETRIEVE iter_max $iter_max;
    RETRIEVE error_min $error_min;
    RETRIEVE error_max $error_max;
    RETRIEVE var1 $var1;
    RETRIEVE var2 $var2;
    ...
    ASSIGN $error ...;
    WHEN : TEST ( ... )
    {
      // Perform needed local actions, if any
    };
    // Too large error.
    WHEN : TEST (>= (abs $error) $error_max) { FAIL; };
    OR
    {
      // Error ok.
      TEST (<= (abs $error) $error_min);
      SUCCEED;
    }
    {
      // Stop if maximum number of iterations reached
      TEST (>= $iter_num $iter_max);
      SUCCEED;
    }
    {
      // Request help from other agents
      // First, set proposal acceptance rules for the negotiation
      EXECUTE SetDecisionParameter "accept" ...;
      EXECUTE SetDecisionParameter "criterium" ...;
      EXECUTE SetDecisionParameter "amount" ...;
      // Calculate local price
      ASSIGN $price  ...;
      EXECUTE SetDecisionParameter "price" $price
      // Perform the actual negotiation
      EXECUTE Negotiate "negotiation_goal_name" $var $error;
      // Increment iteration counter and wait for the next iteration
      EXECUTE Assert "iter_num" (+ $iter_num 1);
      EXECUTE WaitForNextIteration "situation_goal" $var $param;
    };
}
```

Figure 5.3 Pseudocode for an example plan of an initiator in the supervisory control application represented with the plan language of JAM.

```
// Pseudodoce of a plan for a participant in the supervisory
// control application
Plan: {
  GOAL: ACHIEVE negotiation_goal $remote_var $error;
  CONTEXT:
    // This plan applies only for control of a particular variable
    (== $remote_var "remote_var");
  BODY:
    // Get decision variables
    RETRIEVE var1 $var1;
    RETRIEVE var2 $var2;
    ...
    // Calculate new proposal
    ASSIGN $proposed_change ...;
    ASSIGN $price ...;
    // Create a proposal message
    EXECUTE MakeProposalMessage "local_var" $proposed_change $price;
}
```

Figure 5.4 Pseudocode for an example plan of a participant in the supervisory control application represented with the plan language of JAM.

The internal decision-making activity of a process automation agent in the supervisory control application utilises the generic planning capabilities of the agent in order to fulfil its role in the society level search process. The responsibility of a single process automation agent in this search process is to take part in a negotiation at each iteration step during the search if the values of some control variables in its area of responsibility need to be changed. The agent who initiated the search has also the responsibility to decide about its termination. In addition to this the agents also have to manage their run-time data structures and perform their local planning tasks associated to the negotiation process. Although a large part of these activities are mostly implemented by the agent platform for process automation (see Chapter 3), they are used in a way particular to this application.

In the supervisory control application the run-time data structures of process automation agents are needed particularly for storing the state of the iterative search process. The agents need to store necessary data from the previous iteration steps if they need it in the subsequent steps. The run-time data of an initiator includes the initial goal and the number of the iteration step. This data is represented with the goals and assertions run-time data structures of the agent platform (see Chapter 3.2.2). The run-time data of a participant depends on its decision-making rules. Due to the reactive nature of this application the plans and contracts relating to each iteration step are needed to be stored only during that particular step of the iterative search process of the agent society (see Chapter 5.2.1).

In the supervisory control application the local planning activity of a process automation agent is mainly used in a reactive way as a part of the negotiation at one iteration step. The local planning processes of an initiator and a participant agent are different. The input to the local planning activity of an initiator consists of the original goal, the current values of the measurements, the qualitative process model and the number of the iteration step. The outcome from the planning activity is an updated set of values to the control parameters. The input to the local planning activity of a participant consists of the goal received from the initiator, the current values of the measurements and the qualitative process model. The outcome from the planning

activity is a proposal specifying the price of possible control actions. In addition to this reactive negotiation-oriented planning activity there may also be an additional deliberative part in the local planning activities of both the initiator and the participant. These might be needed for sequential local control actions.

The purpose of the negotiation activity in the supervisory control application is to find new values for a set of control variables at each step of the iterative search process. The negotiation is performed according to the Contract Net protocol implemented in the agent platform. The decision-making required at each step of the negotiation protocol and the exact content of the messages exchanged during the negotiation are specific to this type of an application. The decision-making and messages are characterised by their role as a part of an iterative search process. The negotiation activity as it appears in this application is illustrated in Figure 5.5 and explained in more detail below.

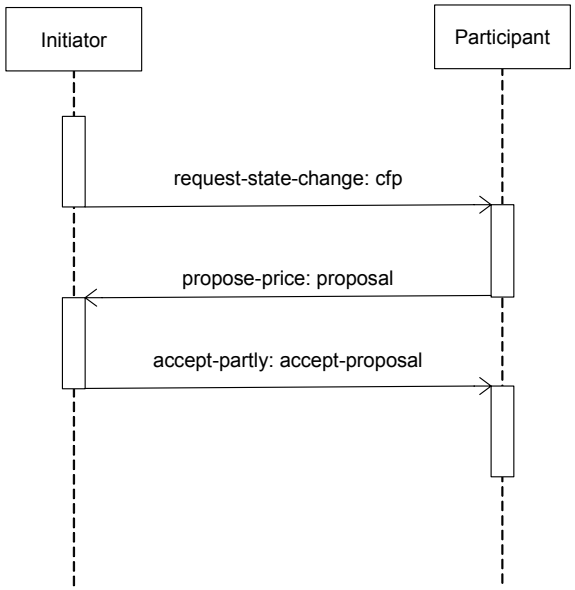


Figure 5.5 Negotiation between two process automation agents in the supervisory control application represented as an AUML sequence diagram.

At the CFP creation stage the decision making task of the initiator agent is to select partners for negotiation and create a goal for changing control variables. The partners are identified using the qualitative process model and the descriptions of agent services and organisational roles at the directory facilitator. The qualitative process model is used to make inferences about which control variables affect the target variable, whereas the agent services are used to determine which agents provide services for controlling these variables. The organisational role is used to find either subordinates or peers of the initiator depending on the application logic. The CFP message contains the description of the control change goal. The goal contains the variable identification and the desired change to it from the viewpoint of the initiator.

At the proposal creation stage the decision making task of the participant is to determine its conditions to affect the control variable in the CFP message. First, the participant identifies which of its control variables can affect the control variable in the CFP. It does this using its qualitative process model or plans. Secondly, the participant estimates its possible effect to the variable in the CFP and determines a cost function for this operation, e.g. in the form of unit price. It does this using some

application specific algorithm specified in a plan. The proposal message contains the maximum value of estimated effect to the variable in the CFP and cost information.

At the answer creation stage the decision-making task of the initiator is to select one or more proposals and decide to which extent to accept the proposed actions. This decision-making logic is meant to be represented as application-specific decision-making rules in the plans. The decision-making logic should balance the changes of control variables in all involved agents. If there are no proposals the initiator concludes that partners are not going to change their control variables at this time and passes this information to its local planning activity. Otherwise accept-proposal messages with specifications of the accepted amounts of proposed changes are sent to the selected participants. Reject-proposal messages are sent to those participants whose proposals are not accepted at all.

At the answer handling stage the participant does not have any real decision making to do. It just needs to calculate the accepted part of its proposed action and execute it.

5.3 Experiments with the supervisory control method

5.3.1 Specification of the experiments

The laboratory test environment with its experimental implementation of the process automation agent platform has been used to test the supervisory control application. A fault recovery application was designed, implemented and tested. The purpose of this experiment was to demonstrate the operation of the application and test its feasibility in simple scenarios. Due to the simplicity and small number of the test scenarios it is not possible to make any thorough assessment of the properties of the control method based on these experiments. The properties of the method are discussed more in Chapter 5.4.

The objective of the fault recovery scenario is to compensate the effects of a fault with respect to the control goals of the test process (see Chapter 3.3) as far as possible. The fault situation is a simulation of a failure of a control valve at the lower part of the tank. Before the fault there is a temperature difference between different parts of the tank. Both of the temperature control loops are operational with distinct setpoints. The objective of process control in this scenario is to minimise the error of water temperature control as expressed in Equation 5.2. Because of the failure of the control valve the temperature control of the lower part of the tank becomes non-functional. Without any compensating control actions the temperature starts deviating from its setpoint and approaches the temperature of the upper part of the tank.

$$e_t = w_1 e_1^2 + w_2 e_2^2 \qquad [5.2]$$

e_t : Total error
e_i : Error at upper or lower part
w_i : Weight of the error at upper or lower part

The fault situation can be compensated with both local recovery actions at the lower part of the tank and remote actions at other parts of the process (see Figure 5.6). At the lower part of the tank the magnetic valves can be used to compensate the water flow of control valves. However, they cannot be used for control due to their

operational limitations. The strength of the water flow through them is not controllable. The temperature control loop at the upper part of the tank can to some extent be used to control the water temperature also at the lower part by the water flow between the parts. The setpoint at the upper part is attempted to be set so that the control errors at both control loops are balanced. This is expected to lead to a better value of the control goal than with only the local recovery actions. The interconnection between the control loops can also to some extent be affected by adjusting the strength of the water circulation by changing the setpoint of the pump.

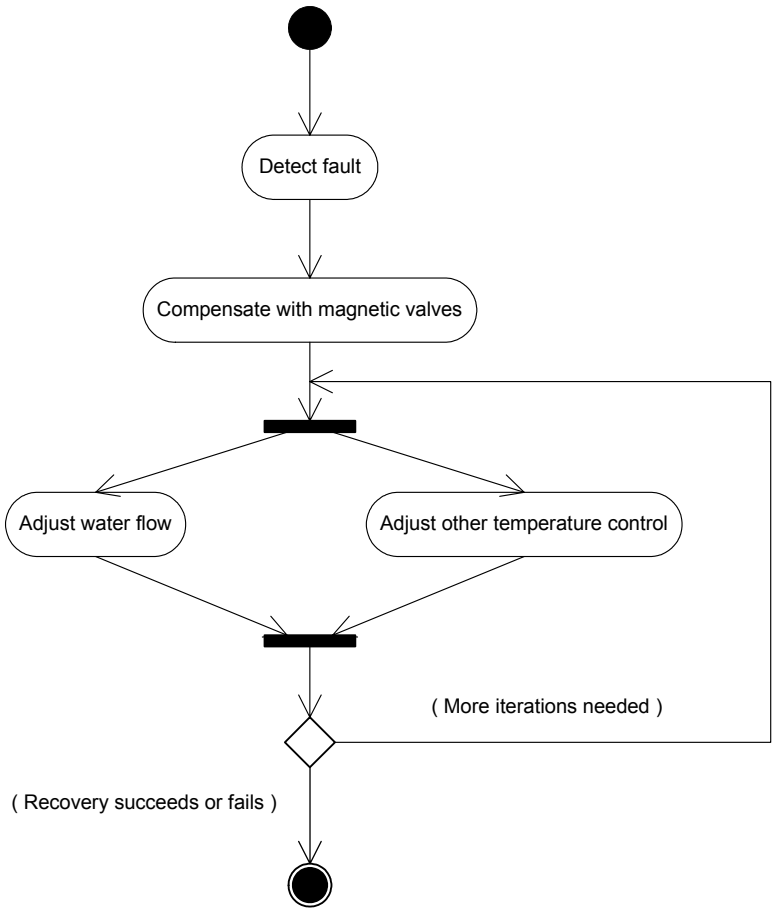


Figure 5.6 Fault recovery scenario in the laboratory test environment represented as an UML activity diagram. One of the control valves is faulty.

5.3.2 Application design for the experiments

The application for the fault recovery scenario was defined as plans that were configured into the plan libraries of the selected agents in the test agent application. For the experiments the plans needed for the initiator role in the fault recovery scenario were defined for the Lower Part Agent. The plans needed for the participant role were defined for the Upper Part Agent and the Pump Agent (see Chapter 3.3.2).

The plans for local recovery actions in the case of control valve failure were defined for the Lower Part Agent. These plans specify a sequence for shutting down non-usable devices and starting compensating water flow with the magnetic valves. The most important one of the local recovery plans is illustrated in Figure 5.7. This plan is fairly similar to the plans representing local actions in the sequential control application (see Chapter 4).

```
// Perform local recovery operations
Plan: {
  GOAL: ACHIEVE local_fault_recovery $var $device;
  CONTEXT:
    // This plan applies only for control of tm2
    (== $var "tm2");
  BODY:
    // Make sure faulted device is shut down
    ACHIEVE shutdown_faulted_device $device;
    // Use the reserve magetic valve
    EXECUTE PlanControlAction "mvsp21" "1";
}
```

Figure 5.7 Pseudocode of the local fault recovery plan of the Lower Part Agent represented with the plan language of JAM.

The Lower Part Agent has a plan for acting as an initiator in a fault recovery search in the case of a control valve failure within its area of responsibility. The plan contains starting the local recovery actions at the first iteration step, decision logic for the negotiations at each step and rules for ending the iterations. The decision logic during the negotiations is to accept help if its price is less than the square of the local error. The iterations are ended after a maximum number of iterations or if the error gets below a minimum value or exceeds a maximum value. The cooperative fault recovery plan of the Lower Part Agent is illustrated in Figure 5.8. The decision parameters used in this plan are explained in Table 5.1.

```
// Perform fault recovery in the case of a device fault at tm2
Plan: {
  GOAL: ACHIEVE fault_recovery $var $device;
  CONTEXT:
    // This plan applies only for control of tm2
    (== $var "tm2");
  BODY:
    // Retrieve decision variables and parameters
    RETRIEVE iter_num $iter_num;
    RETRIEVE iter_max $iter_max;
    RETRIEVE tsp2_weight $tsp2_weight;
    RETRIEVE tm2 $tm2;
    RETRIEVE tsp2 $tsp2;
    ASSIGN $error (- $tsp2 $tm2);
    WHEN : TEST (== $iter_num 0)
    {
      ACHIEVE local_fault_recovery $var $device;
    };
    // Too large error. Recovery fails
    WHEN : TEST (>= (abs $error) $error_max) { FAIL; };
    // At first iteration perform local recovery operations
    OR
    {
      // Error ok. Recovery done
      TEST (<= (abs $error) $error_min);
      SUCCEED;
    }
    {
      // Stop if maximum number of iterations reached
      TEST (>= $iter_num $iter_max);
      SUCCEED;
    }
    {
      // Request help from other agents
      // First, set proposal acceptance rules for the negotiation
      // * accept multiple proposals
      // * accept proposals with lower price than the local price
      // * accept proposals as such
      EXECUTE SetDecisionParameter "accept" "multiple"
      EXECUTE SetDecisionParameter "criterium" "price-min"
      EXECUTE SetDecisionParameter "amount" "all"
      // Calculate local price
      ASSIGN $price (* $tsp2_weight $error);
      ASSIGN $price (* $price $price);
      EXECUTE SetDecisionParameter "price" $price
      // Perform the actual negotiation
      EXECUTE Negotiate "recovery_help" $var $error;
      // Increment iteration counter and wait for the next iteration
      EXECUTE Assert "iter_num" (+ $iter_num 1);
      EXECUTE WaitForNextIteration "fault_revovery" $var $devive;
    };
}
```

Figure 5.8 Pseudocode of the fault recovery search plan of the Lower Part Agent represented with the plan language of JAM.

Table 5.1 Decision parameters of the Lower Part Agent in the fault recovery negotiations.

Parameter	**Explanation**
tsp2_weight	Weight of control error at the lower part of the tank
error_min	Minimum error. Success if went below
error_max	Maximum error. Failure if exceeded
negotiation_wait	Time between negotiations
iteration_max	Maximum number of iterations

The Pump Agent and the Upper Part Agent have plans for helping recovery actions in the case of a fault at the Lower Part Agent. These plans are quite similar. The help they offer is a constant change of their setpoint. The price for the help is the square of the deviation of the new setpoint from the original one. The fault recovery plans of the Pump Agent and the Upper Part Agent are illustrated in Figure 5.9. The decision parameters used in these plans are explained in Table 5.2.

```
// Perform remote help in case of fault recovery
Plan: {
  GOAL: ACHIEVE recovery_help $remote_var $error;
  CONTEXT:
    // This plan applies only for control of tm2
    (== $remote_var "tm2");
  BODY:
    // Get decision variables and parameters
    RETRIEVE tsp1 $tsp1_current;
    RETRIEVE tsp1_original $tsp1_original;
    RETRIEVE tsp1_change $tsp1_change;
    RETRIEVE tsp1_weight $tsp1_weight;
    // Calculate new proposed setpoint and price of the change
    ASSIGN $tsp1_new (- $tsp1_current $tsp1_change);
    ASSIGN $price (* $tsp1_weight (- $tsp1_current $tsp1_new));
    ASSIGN $price (* $price $price);
    // Create a proposal message
    EXECUTE MakeProposalMessage "tm1" $tsp1_change $price;
}

// Perform remote help in case of fault recovery
Plan: {
  GOAL: ACHIEVE recovery_help $remote_var $error;
  CONTEXT:
    // This plan applies only for control of tm2
    (== $remote_var "tm2");
  BODY:
    // Get decision variables and parameters
    RETRIEVE psp $psp_current;
    RETRIEVE psp_original $psp_original;
    RETRIEVE psp_change $psp_change;
    RETRIEVE psp_weight $psp_weight;
    // Calculate new proposed setpoint and price of the change
    ASSIGN $psp_new (- $psp_current $psp_change);
    ASSIGN $price (* $psp_weight (- $psp_original $psp_new));
    ASSIGN $price (* $price $price);
    // Create a proposal message
    EXECUTE MakeProposalMessage "psp" $psp_change $price;
}
```

Figure 5.9 Pseudocode of the fault recovery search plans of the Upper Part Agent and the Pump Agent represented with the plan language of JAM.

Table 5.2 Decision parameters of the Upper Part Agent and the Pump Agent in the fault recovery negotiations.

Parameter	**Explanation**
tsp1_change	Constant change of setpoint at the upper part of the tank
tsp1_weight	Weight of setpoint change at the upper part of the tank
psp_change	Constant change to pump setpoint
psp_weight	Weight of setpoint change for the pump

5.3.3 Results from the experiments

The supervisory control application was tested with three different variations of the fault recovery scenario. The variations demonstrate how local actions of the process automation agents and different negotiation schemes among them affect the fault recovery performance.

In the first variation of the fault recovery scenario the Lower Part agent applies its local recovery plans while other agents remain idle. This variation is presented in order to demonstrate to which extent the local recovery actions alone can compensate the fault. The local recovery actions, i.e. starting a compensating constant water flow from the magnetic valves, are performed in one step after fault detection. There is no search in this scenario. The behaviour of the test process during the first variation of the fault recovery scenario is illustrated in Figure 5.10. Temperature T2 approaches Temperature T1, but a permanent difference remains between them because of the compensating water flow from the magnetic valves. The control errors are illustrated in Figure 5.11.

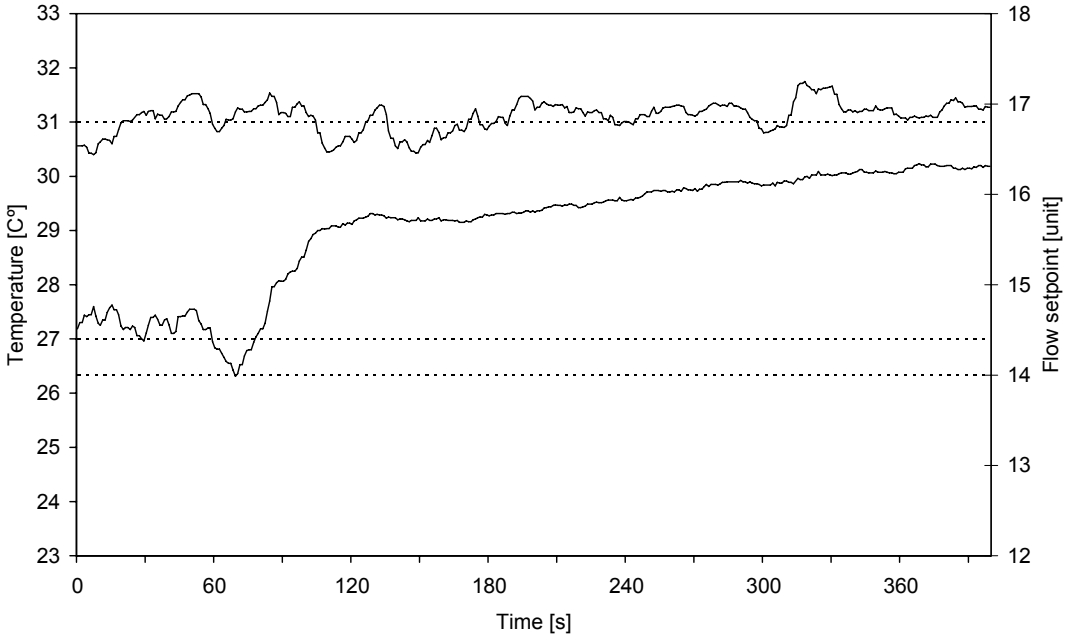


Figure 5.10 Selected process variables during the first variation of the fault recovery scenario. In this variation, Lower Part Agent applies its local recovery actions. The solid lines represent the temperature measurements T1 (above) and T2 (below). The dotted lines represent setpoints of the temperatures (above and middle) and the water flow (below).

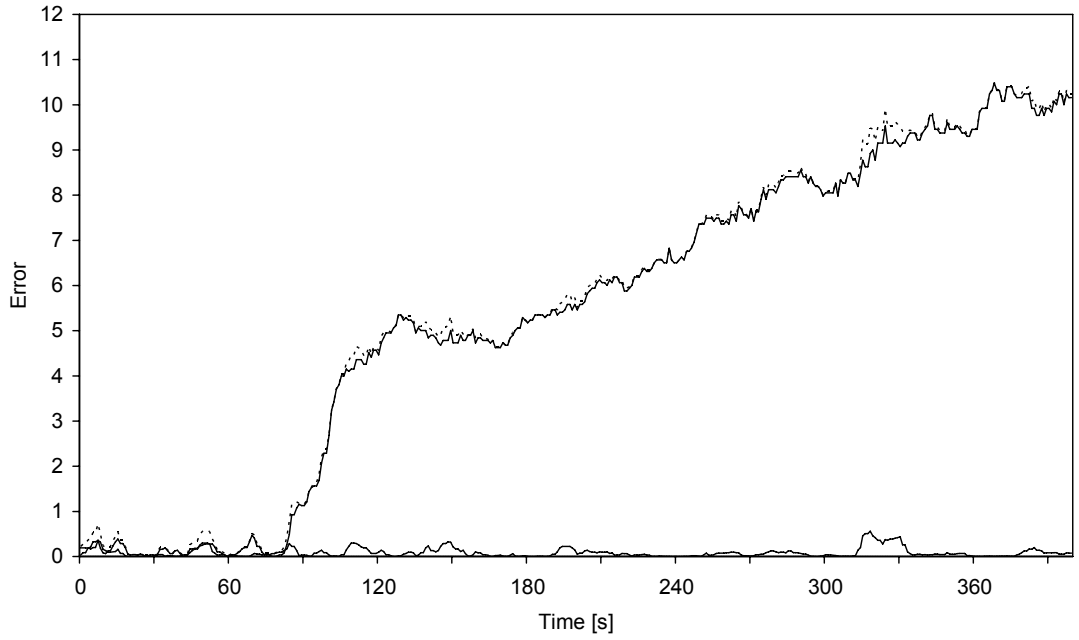


Figure 5.11 Control errors during the first variation of the fault recovery scenario. The solid lines represent the control errors at the upper and the lower parts of the tank. The dotted line represents the total error.

The second variation of the fault recovery scenario introduces a one-to-one negotiation between the Lower Part agent and the Upper Part agent in addition to the local actions of the previous one. The cooperation between the two negotiating agents during the iterative search process of the fault recovery is illustrated in Table 5.3. During the negotiations the Upper Part Agent gradually changes its setpoint until the control error is distributed evenly between it and the Lower Part Agent. The behaviour of the test process during the second variation of the fault recovery scenario is illustrated in Figure 5.12. Also in this variation the Temperature T2 is diverging from its setpoint and approaches T1. However, it does not diverge from its setpoint as much as in the previous variation. The control errors are illustrated in Figure 5.13. The performance of the fault recovery is clearly better than in the first variation of the scenario.

Table 5.3 Message exchange between the two negotiating agents during the iterative search process in the second variation of the fault recovery scenario.

No	Sender	Receiver	Message type	Message content
1	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -0,64
2	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 0.06
3	Lower Part	Upper Part	accept_proposal	
4	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,64
5	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 0.25
6	Lower Part	Upper Part	accept_proposal	
7	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -2,14
8	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 0.56
9	Lower Part	Upper Part	accept_proposal	
10	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -2,18
11	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 1.0
12	Lower Part	Upper Part	accept_proposal	
13	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -2,08
14	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 1.56
15	Lower Part	Upper Part	accept_proposal	
16	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -2,10
17	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 2.25
18	Lower Part	Upper Part	accept_proposal	
19	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -2,08
20	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 3.06
21	Lower Part	Upper Part	accept_proposal	
22	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,95
23	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 4.0
24	Lower Part	Upper Part	reject_proposal	
25	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,71
26	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 4.0
27	Lower Part	Upper Part	reject_proposal	
28	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,59
29	Upper Part	Lower Part	proposal	measure: tm1, amount: 0.25, price: 4.0
30	Lower Part	Upper Part	reject_proposal	

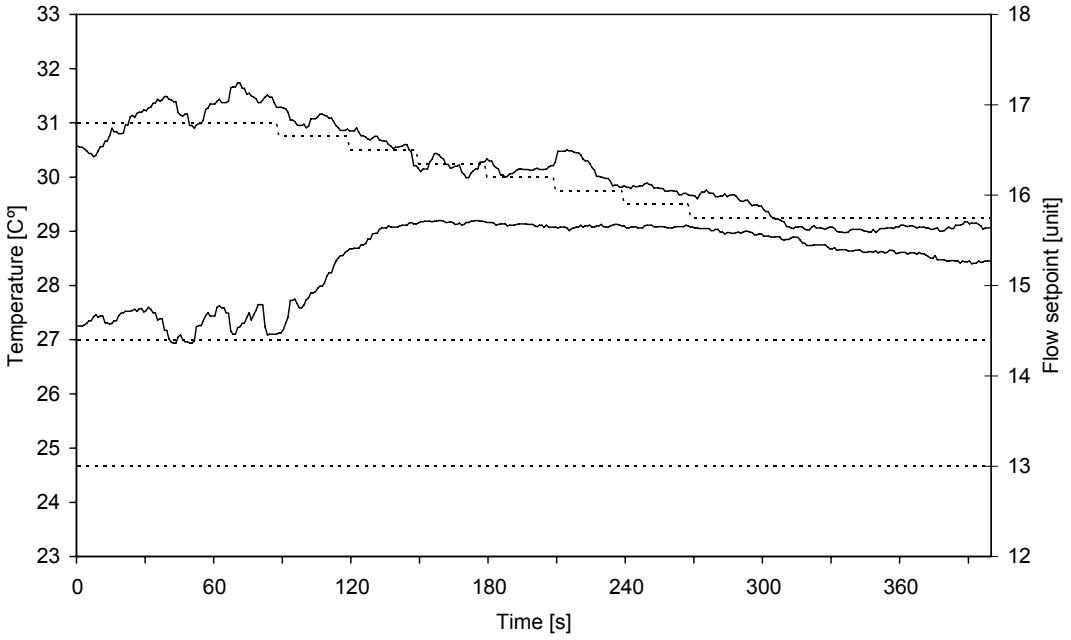


Figure 5.12 Selected process variables during the second variation of the fault recovery scenario. In this variation, Lower Part Agent applies its local recovery actions and cooperates with Upper Part Agent. The solid lines represent the temperature measurements T1 (above) and T2 (below). The dotted lines represent setpoints of the temperatures (above and middle) and the water flow (below).

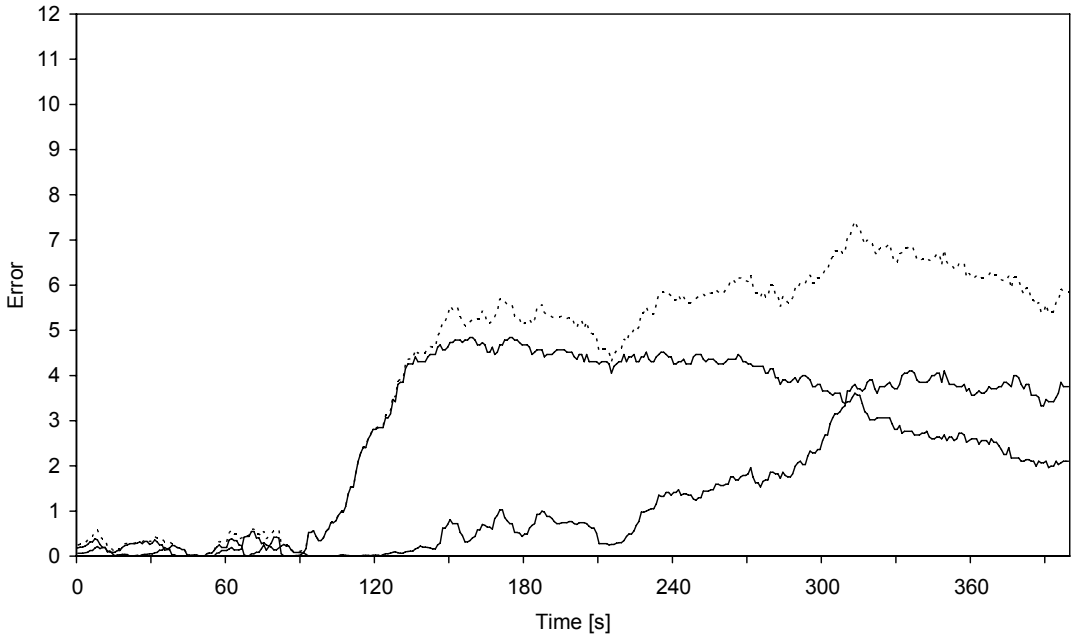


Figure 5.13 Control errors during the second variation of the fault recovery scenario. The solid lines represent the control errors at the upper and the lower parts of the tank. The dotted line represents the total error.

The third variation of the fault recovery scenario adds a further negotiation relationship between the Lower Part Agent and the Pump Agent in addition to the second variation. The cooperation among the three negotiating agents during the

iterative search process of the fault recovery is illustrated in Table 5.4. Both the Upper Part Agent and the Pump Agent negotiate with the Lower Part Agent in quite a similar way. The behaviour of the test process during the third variation of the fault recovery scenario is illustrated in Figure 5.14. The development of the temperatures in this variation is quite similar to the previous variation without the pump. In these tests the effect of the strength of the water flow on the temperatures is quite weak. The control errors are illustrated in Figure 5.15. The performance of the fault recovery is roughly the same than in the second variation of the scenario.

Table 5.4 Part of the message exchange among the three negotiating agents during the iterative search process in the third variation of the fault recovery scenario (thirty first messages).

No	Sender	Receiver	Message type	Message content
1	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -0,05
2	Lower Part	Pump	cfp	goal: recovery_help tm2 -0,05
3	Upper Part	Lower Part	proposal	measure: tm1 amount: 0.25 price: 0.06
4	Pump	Lower Part	proposal	measure: psp amount: 0.25 price: 0.25
5	Lower Part	Upper Part	reject_proposal	
6	Lower Part	Pump	reject_proposal	
7	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,51
8	Lower Part	Pump	cfp	goal: recovery_help tm2 -1,51
9	Pump	Lower Part	proposal	measure: psp amount: 0.25 price: 0.25
10	Upper Part	Lower Part	proposal	measure: tm1 amount: 0.25 price: 0.06
11	Lower Part	Upper Part	accept_proposal	
12	Lower Part	Pump	accept_proposal	
13	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,82
14	Lower Part	Pump	cfp	goal: recovery_help tm2 -1,82
15	Pump	Lower Part	proposal	measure: psp amount: 0.25 price: 1.0
16	Upper Part	Lower Part	proposal	measure: tm1 amount: 0.25 price: 0.25
17	Lower Part	Upper Part	accept_proposal	
18	Lower Part	Pump	accept_proposal	
19	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,84
20	Lower Part	Pump	cfp	goal: recovery_help tm2 -1,84
21	Pump	Lower Part	proposal	measure: psp amount: 0.25 price: 2.25
22	Upper Part	Lower Part	proposal	measure: tm1 amount: 0.25 price: 0.56
23	Lower Part	Upper Part	accept_proposal	
24	Lower Part	Pump	accept_proposal	
25	Lower Part	Upper Part	cfp	goal: recovery_help tm2 -1,64
26	Lower Part	Pump	cfp	goal: recovery_help tm2 -1,64
27	Pump	Lower Part	proposal	measure: psp amount: 0.25 price: 4.0
28	Upper Part	Lower Part	proposal	measure: tm1 amount: 0.25 price: 1.0
29	Lower Part	Upper Part	accept_proposal	
30	Lower Part	Pump	reject_proposal	
31	…			

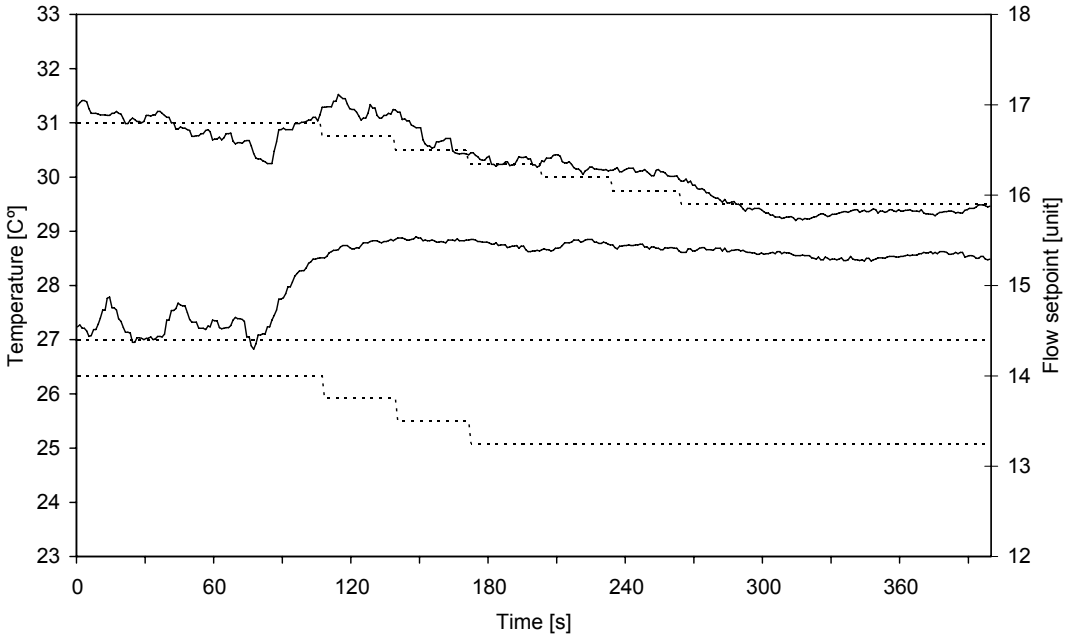


Figure 5.14 Selected process variables during the third variation of the fault recovery scenario. In this variation, Lower Part Agent applies its local recovery actions and cooperates with both Upper Part Agent and Pump Agent. The solid lines represent the temperature measurements T1 (above) and T2 (below). The dotted lines represent setpoints of the temperatures (above and middle) and the water flow (below).

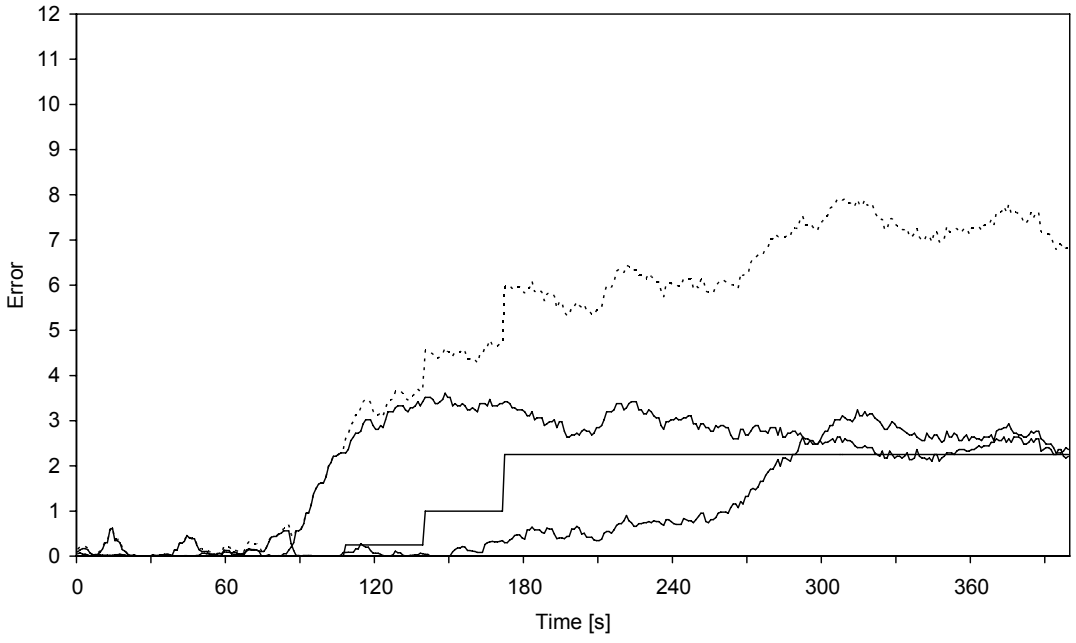


Figure 5.15 Control errors during the third variation of the fault recovery scenario. The solid lines represent the control errors at the upper and the lower parts of the tank and the pump. The dotted line represents the total error.

5.4 Discussion

The specification of the supervisory control application is assessed in here by comparing it to related reported supervisory control approaches and discussing its properties. Similarities and differences to the references are characterised and justifications to the adopted design choices are presented. The effect of these choices on the automation system properties is discussed and compared to the references. Finally, conclusions are drawn based on this study and open questions concerning the research topic are outlined.

5.4.1 Design choices in the specification

The society model of the supervisory control application has important common features with previously reported research (see Chapter 2.3). Important related research efforts in this context include the studies by Ygge (Ygge 1998, Ygge and Akkermans 1999), van Breemen (van Breemen and de Vries 2001, van Breemen 2001). The main similarity of the presented agent society model with the references is the existence of one coordinator that has the responsibility to coordinate the operation of other controllers. This idea of a coordinator is not a new one and has been presented earlier e.g. in the studies of so-called hierarchical multi-level systems (Mesarovics at al. 1970). However, the specified agent society model also has some differences from the references. One difference is that the role of coordinator is not fixed and is allowed to change depending on the situation. This is assumed to enhance system responsiveness. Other differences include the Contract Net-based negotiation mechanism and the goal-centered ontology. In the presented specification these are provided by the agent platform and shared with other applications as general models of cooperation.

The agent model of the supervisory control application does not have many references in the previously reported research (see Chapter 2.3). MACIF (van Breemen 2001) is one of the few reported respective models. The specified agent model and MACIF are quite different. The main difference is that the presented specification follows the BDI-model of internal agent architecture whereas the agents of MACIF may be characterised as wrappers of controllers containing inputs, outputs, control algorithm and an interface for activation of the agent. In the presented specification the BDI-model is provided by the agent platform and shared with other applications as a general model of agent operation.

5.4.2 Effects on the properties of automation

The performance of the supervisory control application depends on both its properties as a distributed search method and its properties as a control application. The distributed search method needs to solve the coordination problem of distributed search (see Chapter 2.2.3) in such a way that it fulfils the requirements of the control application. The important property of the method in this sense is optimality of the distributed search algorithm. This algorithm is partially defined in the presented specification of the application and partially in the plans. The plans contain the actual decision logic of how to calculate new values for the control variables at each iteration of the search. The performance of the application can be analysed only when this decision logic is known. The fault recovery test application defines a simple search strategy as a demonstration. In this experimental application the search is based

on process-specific heuristics whose parameters are tuned based on experiments with the process. Feedback from the process measurements is utilised only as the search termination criterion.

The specification of the supervisory control application is expected to facilitate the reconfigurability property of process automation systems. Similarly to the sequential control case (Chapter 4) also with this application the effect on the reconfigurability property is assumed to originate from the combination of the mechanisms of the agent platform and the application. The used mechanisms of the platform include the qualitative process model, the directory facilitator, the Contract Net based negotiation and the internal planning activity of the agents with its modification for cyclic operation. The supervisory control application adds the distributed iterative search process on top of them. With the combination of these mechanisms it is possible to modify the values of supervisory control variables if the configuration changes require it. The mechanism can react to changes in the existence and capabilities of the agents. These changes reflect changes in the underlying automation system, e.g. device replacement and maintenance operations. The agents can e.g. balance the values of set-points in various controllers or apply different control policies. The possibility of reconfiguration is limited by the capability of the lower-level automation to provide alternative means to gain the required control operations, e.g. some control variables are controllable only by one particular device. The references do not include any approaches that would have all the same mechanisms for reconfiguration. Maybe the most significant difference to the references is the possibility to include also sequential control operations in the reconfiguration process together with negotiation about the control variables.

The specification of the supervisory control application is also expected to facilitate enhancement of the flexibility and responsiveness properties of process automation systems. Similarly to the sequential control case (Chapter 4) also with this application the intended means for this purpose is to design control applications that can handle planned or unplanned operational change situations and conform to the presented model of distributed search. The distributed iterative search process can be used to find new values for a set of control variables. If operational change situations can be handled through this iterative search process then the specification can facilitate flexibility and responsiveness. The situations intended to be handled have the characteristic that values of several inter-dependent supervisory control variables have to be balanced, e.g. when changing the set-points of inter-related PID-controllers after a significant change in the process conditions. The references utilise different methods but they share a similar objective with respect to flexibility and responsiveness.

The specification of the supervisory control application shows how the application design model of the agent platform can be used for developing supervisory control applications. The applications are designed by defining the decision logic for changing values of some supervisory control variables after a negotiation between a coordinator and a set of other agents. This kind of an approach for designing supervisory control applications in automation is different from the more traditional command-based coordination, although the idea about negotiation as a coordination method in continuous control has been proposed also in earlier studies (Ygge and Akkermans 1999). The presented model of application development proposes plans as a method for representing negotiation logic and goal and precondition information as mechanisms for determining their applicability. This model is fairly general and

allows design of various different negotiation decision logics. It does not define how to calculate values for the control variables.

5.4.3 Conclusions and open questions

The specification of the supervisory control application may be characterised as an aggregate design and the experimental fault recovery application as a demonstrating prototype. However, it is possible to recognise the basic advantages and limitations of the application. Analogously to the sequential control application the main advantage of this application is the enabled positive effect on the adaptation properties of automation. Another advantage is the possibility to implement the application using a platform which is designed to be able to run also sequential control functions of process automation. The main limitation of the application concerns the distributed search process used in it. The performance of the search process depends on the decision logic of the agents. Designing such decision logic for various situations is not an easy task. Another limitation is the response time of the negotiation at each iteration, which restricts the control cycle of this application. Also the characteristics of the underlying automation system and the controlled process affect the usefulness of the supervisory control application. The possible lack of redundancy in them, i.e. alternative ways to affect objective control variables, restricts the adaptation properties also in this application. The significance of the experiments described in this study is that they are able to demonstrate the characteristics of distributed search process and the design of plans for it in simple test scenarios.

The presented form of the specification leaves important open questions concerning the utilisation of MAS in supervisory control applications in process automation. Maybe the most important open question is if the presented aggregate specification of distributed search is feasible in a more general case and how to make it more detailed so it could be better evaluated. These questions could be answered by studying the decision logic of the agents during the distributed iterative search process more closely. Particularly the decision logic of the coordinator with its need to handle possible inter-dependencies between the actions of separate agents is an important one. It would also be worthwhile to study other control properties of this application than optimality, e.g. stability and robustness. Test scenarios with more challenging control tasks could be useful, e.g. in the form of simulations.

6 Discussion and conclusions

6.1 Discussion

The specifications of the agent platform for process automation and the experimental control applications presented in this thesis are discussed here by comparing them to similar reported approaches and assessing their effect on the properties of automation. This discussion summarises the most important issues of earlier discussions in Chapters 3, 4 and 5. Differences to the references are characterised and justifications for the adopted design choices presented. Also the knowledge obtained about the effect of the approach to the properties of automation is discussed.

The specifications presented in this thesis share many similarities with earlier studies, particularly the ones developed within the HMS consortium. The main similarities include the role of agents as a supervisory control system, hierarchical and goal-oriented agent organisation similar to a holarchy, Contract Net as a negotiation mechanism, BDI-agent model, usage of a directory facilitator and conformance to the FIPA standard. However, the presented specifications contain also some important differences to the earlier research. Whereas the main part of the previous research has focused on discrete manufacturing, the application domain in this thesis is process automation. When compared to other research in process automation, a major contribution in this thesis is a common BDI-based model for both sequential and continuous control. For this purpose the BDI-model is extended with a possibility for cyclic operation. Goals are used as the main data structure and distributed search and planning processes as the main problem-solving methods in the coordination among the agents. Another important new feature in the presented specifications is the usage of qualitative reasoning for identification of the coordination needs. A further difference to the references is a system architecture utilizing both vertical and horizontal coordination channels.

The performance of the control applications specified in this thesis depends on both their properties as distributed problem-solving methods and their properties as control applications. The distributed problem-solving methods are required to solve the coordination problem of the distributed applications in such a way that they can fulfil their requirements as control applications. The applications specified in this thesis are either based on distributed planning or distributed iterative search. However, the specifications contain only aggregate designs for general cases. Detailed designs are presented for the experimental test applications. Because of this, it is not possible to make a general performance assessment of the control applications. Also in the related research the performance assessment of this kind of applications has been limited. The performance of the applications depends particularly on the applied distributed search strategy in the sequential control application, and the decision logic of the agents in the supervisory continuous control application. In the latter application also the response time of the negotiation may limit its applicability.

The specifications in this thesis describe a platform with mechanisms for enhanced reconfigurability of process automation systems and applications designed to utilise these mechanisms. The autonomy property of the agents, utilisation of a directory facilitator, distributed search and planning as problem-solving methods and Contract Net as a negotiation mechanism are expected to enable process automation agents to operate semi-independently and coordinate configuration to changes with other

agents. These mechanisms are assumed to be able to handle changes in the existence and capabilities of agents and in the lower-level automation system that the agents are supervising. However, in order these mechanisms to be useful they have to be used adequately in applications. The applications need to contain logic for changing the control parameters of the lower level automation and coordinating these changes with other agents. However, the benefit of the reconfiguration capability is limited if there is lack of redundancy in the lower-level automation system and the controlled process. The effect on reconfigurability in this study is quite similar to some references (Tichy at al. 2002). The main difference is the possibility to include both sequential and continuous control operations in the reconfiguration process.

The applications specified in this thesis demonstrate the capabilities of MAS-based problem-solving methods in the design of applications which are expected to enhance the flexibility and responsiveness properties of process automation systems. Applications designed as distributed planning of control sequences and distributed iterative search of values for supervisory control variables are proposed as means to handle both planned and unplanned operational change situations. A contribution in this thesis is that both sequential and continuous control operations are combined in the MAS in order to facilitate the flexibility and responsiveness properties. The distributed planning process is able to create new sequences as a combination of the sequences wrapped in the plans. The distributed iterative search process can be used to find new values for a set of control variables. If operational change situations can be handled with these methods then the applications can facilitate flexibility and responsiveness. Concerning the sequential case the effect on the flexibility and responsiveness properties is similar to some references (Tichy at al. 2002).

The specifications presented in this thesis also define a model for developing applications. According to this model, higher-level control applications operating on top of an ordinary process automation system are designed as distributed planning and search applications. The applications are programmed as plans that are used during the planning and search processes. This model of designing applications for process automation is quite new although similar approaches have been proposed in references (Tichy at al. 2002, Ingrand et al. 1992). There is not much experience about how well this works. A particular aspect in this research is that the same MAS-based application development model is applied both to sequential and supervisory continuous control operations.

6.2 Conclusions

The specifications, experiments and discussions presented in this thesis allow outlining some conclusions about the applicability of MAS in process automation. The conclusions are divided here according to the research objectives of this thesis (see Chapter 1.3) into four parts concerning the specifications of the agent platform and the two applications and the assessment of their properties.

The first research objective of this thesis was to make a specification of an agent platform for process automation. The specification outlines the BDI-agent model with cyclic operation, qualitative process models, agent organisation with just one agent type and utilisation of coordination among peers as particular requirements or useful features of an agent platform, which otherwise is quite similar to referenced HMS (Tichy at al. 2002, Maturana at al. 2002, 2003, Heikkilä at al. 1999). The experimental applications developed in this thesis demonstrate the feasibility of the platform as an

application development tool. Mechanisms supporting useful adaptation properties can be identified in the specification. Based on this study it seems reasonable to state that when applying MAS to process automation it is useful for an agent platform to contain at least the features specified in this study.

The specification of the sequential control application using the agent platform for process automation is a contribution to the second research objective of this thesis. The specification outlines an application with distributed planning as a problem-solving method. The experimental applications indicate the feasibility of the design in the presented test scenarios. The possible benefit of distributed planning for the adaptation properties can be identified. Some other studies have also developed quite similar applications for sequential control (Tichy at al. 2002, Maturana at al. 2002, 2003). However, there are important limitations in the presented approach. The essential limitation of the approach is the lack of knowledge about the performance of the distributed planning algorithms in more complicated test cases. As a conclusion about this application it seems reasonable to state that if a proper search strategy can be defined for an application then the specified mechanisms are applicable and useful.

The specification of the supervisory control application using the agent platform for process automation is another contribution to the second research objective of this thesis. The specification outlines an application with distributed iterative search as a problem-solving method. Also in this case the experimental applications can indicate the feasibility of the design in the presented test scenarios. Again, the possible benefit of distributed iterative search for the adaptation properties can be identified. Although some other studies have also utilised similar mechanisms a direct counterpart of this application could not be identified. However, there are important limitations also in this application. The essential limitation of this application is the difficulty to design a decision logic that would guarantee adequate performance of the distributed search process also in more complicated test cases. As a conclusion about this application it seems reasonable to state that if a proper decision logic can be defined for the various agents in an application and the required response time is not too short then the specified mechanisms are applicable and useful.

The third research objective of this thesis was to assess the specifications of the agent platform and the control applications. This has already been done in several parts this thesis describing the experiments (chapters 4.3, 5.3) and discussions (chapters 3.4, 4.4, 5.4 and 6.1). The performance of the applications for control purposes cannot be assessed in a general case because of the aggregate nature of the specifications. The specifications contain mechanisms that can be argued to enhance the adaptation properties of process automation. However, this benefit is limited by the properties of the underlying automation system and the possible difficulty of the application design. There is not much experience from the plan-based application design model. As a conclusion about the assessment one could state that the specifications contain promising features particularly concerning the adaptation properties but otherwise the knowledge about the assessed properties is limited.

6.3 Further research

The results of this thesis allow proposing topics for further research for the application of MAS in process automation. The topics concern the open questions that have not found their answers in this thesis or in related research. The topics relate to the specification of the agent platform, the applications and evaluation of their properties.

The first topic for further research is to consider the specification of the agent platform from the viewpoint of other functions of process automation than control. The objective of such research is proposed to be a more general agent platform for process automation, which is able to run also other types of applications, e.g. process monitoring, condition monitoring and diagnostics, in addition to control applications. The proposed method for this research is to extend and possibly redesign the agent platform based on the requirements of new applications. Research concerning this topic has already started (Pirttioja et al. 2004).

The second topic for further research is the development of more advanced distributed planning and search methods for the applications studied in this thesis. The objective of this research would be to find out how the different features of the distributed planning and iterative search methods affect the properties of MAS-based process automation. This question could be studied both with simulation studies and analysis of the search strategies and decision logic from the viewpoints of computer science (Yokoo and Ishida 1999), optimisation (Bazaraa et al. 1993) and systems theory (Findeisen 1980).

The third topic for further research is a more thorough assessment of the platform and the applications. The properties to be evaluated are the same ones as already studied in this research, in other words the properties relating to adaptation, i.e. reconfigurability, flexibility and responsiveness, the properties relating to performance as a distributed control system, i.e. stability, robustness and scalability, and the properties as an application design model. Simulation studies are proposed as a suitable method for studying the first two sets of properties. With simulations it would be possible to study larger and more complicated control problems and more diverse test scenarios than what was possible to do within this work. Some research has already been initiated in this area (Maturana et al. 2005b). Development of prototype applications and their evaluation by process automation application programmers are proposed as suitable methods for assessing the application development model.

References

Adelsberger, H. H., Conen, W. 2000. Economic Coordination Mechanisms for Holonic Multi Agent Systems, Proceedings of the 11th International Conference on Database and Expert Systems Applications (DEXA 2000), London, UK.

Arai, T., Aiyama, Y., Sugi, M., Ota, J. 2001. Holonic Assembly System with Plug and Produce. Computers in Industry, Elsevier, Vol. 46, pp. 289-299.

Bratman, M. E., Israel, D. J., Pollack, M. E. 1988. Plans and Resource Bounded Practical Reasoning. Computational Intelligence, Vol. 4, pp. 349-355.

Bazaraa, M. S., Hanif, D. S., Shetty, C. M. 1993. Nonlinear Programming, Theory and Algorithms, 2nd ed., John Wiley & Sons, Inc.

Brennan, R. W., Fletcher, M., Norrie, D. H. 2002a. A Holonic approach to reconfiguring real-time distributed control systems. In: Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.) Multi-Agent Systems and Applications II, Springer, Germany, pp. 323 – 335.

Brennan, R. W., Fletcher, M., Norrie, D. H. 2002b. An Agent-Based Approach to Reconfiguration of Real-Time Distributed Control Systems, IEEE Transactions on Robotics and Automation, Vol. 18, No. 4, pp. 444-451.

Brooks, R, A. 1991. Intelligence without Reason, Proceedings of the 12th International Joint Conference on Artificial Intelligence, pages 569–595, Sydney, Australia, pp. 569 – 595.

Bussmann, S., Jennings, N. R., Wooldridge, M. 2004. Multiagent Systems for Manufacturing Control: A Design Methodology, Springer.

Bussmann, S., Schild, K. 2001. An Agent-based Approach to the Control of Flexible Production Systems, Proceedings of the 8th IEEE International Conference on Emergent Technologies and Factory Automation (ETFA 2001), Vol. 2, Antibes Juan-les-pins, France, pp. 481-488.

Castro, J., Kolp, M., Mylopoulos, J. 2002. Towards Requirements-Driven Information Systems Engineering: The Tropos Project, Université catholique de Louvain, Institut d'administration et de gestion, Working paper 31/02.

Chakraborty, S. 2003. Agent-Based Approach to Fault Recovery in a Process Automation System, Master's Thesis, Helsinki University of Technology and Technische Universität Darmstadt.

Chiu, S., Chen, Y.-L., Provan, G., Maturana, F., Staron, R., Hall, K. 2003. Distributed Diagnostics & Reconfiguration for Shipboard Chilled Water System, 13th International Ship Control Systems Symposium (SCSS), Orlando, FL.

Chokshi, N. N., McFarlane, D. C. 2002. Rationales for Holonic Applications in Chemical Process Industry. In: Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.). Multi-Agent Systems and Applications II, Springer, Germany, pp. 323-335.

Chokshi, N. N. 2004. Holonic Process Control: A Distributed, Collaborative Approach to the Control of Chemical Process Operations, Doctorate thesis, University of Cambridge, Churchill College.

Cockburn, D., Jennings, N. R. 1995. ARCHON: A Distributed Artificial Intelligence System for Industrial Applications, In: O'Hare, G. M. P., Jennings, N. R. (eds.) Foundations of Distributed Artificial Intelligence. Wiley & Sons.

Deen, S. (ed.) 2003. Agent-Based Manufacturing, Advances in the Holonic Approach. Springer.

Deen, S. M., Fletcher, M. 2000. Temperature equilibrium in multi-agent systems, Proceedings of the 11th International Conference on Database and Expert Systems Applications (DEXA 2000), London, UK.

de Kleer, J., Braun, J. 1984. A Qualitative Physics Based on Confluences, Artificial Intelligence, Vol. 24, pp. 7-84.

desJardins, M., Durfee, E., Ortiz, C., Wolverton, M. 1999. A Survey of Research in Distributed Continual Planning, AI Magazine, Winter, pp. 13-22.

Duffie, N. A., Chitturi, R., Mou, J. 1987. Fault-tolerant heterarchical control of heterogeneous manufacturing system entities, Journal of Manufacturing Systems, Vol. 7, pp. 315-327, 1987.

Durfee, E. H. 1999. Distributed Problem Solving and Planning, In: Weiss, G. (ed.) Multiagent Systems. MIT Press, pp. 121-164.

Fajt, M. 2003. Information agents in process automation, Master's Thesis, Helsinki University of Technology, Automation Technology Laboratory.

Ferber, J. 1999. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley Pub. Co.

FIPA. 2002a. FIPA Abstract Architecture Specification, FIPA Standard SC00001.

FIPA 2002b. FIPA ACL Message Structure Specification, FIPA Standard SC00061.

FIPA. 2002c. FIPA Communicative Act Library Specification, FIPA Standard SC00037.

FIPA. 2002d. FIPA Contract Net Interaction Protocol Specification, FIPA Standard SC00029.

FIPA. 2002e. FIPA Request Interaction Protocol Specification, FIPA Standard SC00026.

FIPA. 2004. FIPA Agent Management Specification, FIPA Standard SC00023.

FIPA. 2005. FIPA homepage, http://www.fipa.org [referenced 31.10.2005]

FIPA-OS. 2005. http://sourceforge.net/projects/fipa-os [referenced 31.10.2005]

Fisher, K., Muller, J. P., Pischel, M. 1994. Unifying Control in a Layerred Agent Architecture, Tchnical Report TM-94-05, DFKI GmbH.

Flake, S., Geiger, C., Kustler, J. M. 2001. Towards UML-based Analysis and Design of Multi-Agent Systems, Proceedings of the International NAISO Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems (ENAIS 2001), Dubai.

Fletcher, M., Brennan, R. W. 2002. Designing an Integrated Holonic Scheduler with JACK. Proceedings of the 13th International Workshop on Database and Expert Systems Applications, (DEXA 2002).

Fletcher, M., Brennan, R. W., Norrie, D. H., Fleetwood, M. 2001. Reconfiguration Processes in a Holonic Sawmill, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Tucson, USA, pp. 158-163.

Fletcher, M., Deen, S.M. 2001. Fault-Tolerant Holonic Manufacturing Systems, Concurrency and Computation Practice & Experience, Vol.13, No.1, pp.43-70.

Forbus, K. D. 1996. Qualitative reasoning, In: Tucker (ed.), The Computer Science and Engineering Handbook, CRC Press.

GE Fanuc Automation. 2005. GE Fanuc Automation homepage. http://www.gefanuc.com [referenced 31.10.2005]

Georgeff, M., Pell, B., Pollack, M., Tarnbe, M., Wooldridge, M. 1999. The Belief-Desire-Intention Model of Agency, In: Muller, J. P., Singh, M., Rao, A. (eds.) Intelligent Agents V, Lecture Notes in AI, Vol. 1365, Springer-Verlag.

Hall, K, H., Staron, R. J., Vrba, P. 2005. Experience with Holonic and Agent-Based Control systems and Their Adoption by Industry, Proceedings of the 2nd International Conference on Applications of Holonic and Multi-Agent Systems (HoloMAS 2005), Copenhagen, Denmark.

Heck, F., Laengle, T., Wörn, H. 1998. A Multi-Agent Based Monitoring and Diagnostics System for Industrial Components, Proceedings of the DX´98, pp. 63-69.

Heikkilä, T., Kollingbaum, M., Valckenaers, P., Bluemink, G.-J. 1999. manAge: An Agent Architecture for Manufacturing Control, Computers in Industry, Vol. 46, pp. 315-331.

Huber, M. J. 1999. JAM Agents in a Nutshell, Version 0.61+0.79i, Intelligent Reasoning systems.

Huber, M. J. 2000. JAM: A BDI-theoretic Mobile Agent Architecture, AgentLink News, No. 5, pp. 3-6.

Huhns, M. N., Stephens, L. M. 1999. Multiagent Systems and Societies of Agents, In: Weiss, G. (ed.) Multiagent Systems. MIT Press, pp. 79-120.

Ingrand, F., Georgeff, M., Rao, A. 1992. An Architecture for Real-Time Reasoning and System Control, IEEE Expert, Vol. 7, No. 6, pp. 34-44.

ISA. 2000. Enterprise - control system integration, Part 3: Models of manufacturing operations. Draft 3.

Iwasaki, Y. 1997. Real World Applications of Qualitative Reasoning, IEEE Expert, Vol. 12, No. 3, pp. 16-21.

JADE. 2005. JADE homepage. http://jade.tilab.com [referenced 31.10.2005]

Jadex. 2005. Jadex homepage. http://vsis-www.informatik.uni-hamburg.de/projects/jadex [referenced 31.10.2005]

Jennings, N. R. 2000. On Agent-Based Software Engineering, Artificial Intelligence, No. 117, pp. 277-296.

Jennings, N. R. 2001. An agent-based approach for building complex software systems. Communications of the ACM, Vol. 44, No. 4, pp. 35-41.

Jennings, N. R., Bussmann, S. 2003. Agent-Based Control Systems, IEEE Control Systems Magazine, pp. 61-73.

Jennings, N.R., Mamdani, E. H., Corera, J. M., Laresgoiti, I., Perriollat, F., Skarek, P., Varga, L. Z. 1996. Using Archon to Develop Real-World DAI Applications, Part 1, IEEE Expert: Intelligent Systems and Their Applications, Vol. 11, No. 6, pp. 64-70.

Jennings, N. R., Wooldridge, M. 2000. Agent-Oriented Software Engineering, Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (MAAMAW-99).

Jess. 2005. Jess homepage. http://herzberg.ca.sandia.gov/jess [referenced 31.10.2005]

Karhela, T. 2002. A Software Architecture for Configuration and Usage of Process Simulation Models, Software Component Technology and XML-based Approach, VTT Publications No. 479, Espoo, Finland.

Krebsbach, K., D., Musliner, D. J. 1998. Applying a Procedural and Reactive Approach to Abnormal Situations in Refinery Control. Proceedings of the Conference on Foundations of Computer-Aided Process Operations (FOCAPO).

Kuikka, S. 1999. A batch process management framework, Domain-specific, design pattern and software component based approach. VTT Publications No. 398, Espoo, Finland.

Labrou, Y., T. Finin, T., Peng, Y. 1999. Agent Communication Languages: The Current Landscape, IEEE Intelligent Systems, Vol. 14, No. 2, pp. 45-52.

Luck, M., Ashri, R., D'Iverno, M. 2004. Agent-Based Software Development, Artech House, Inc.

Luder, A., Klostermeyer, A., Peschke, J., Bratoukhine, A., Sauter, T. 2005. Distributed Automation: PABADIS versus HMS, IEEE Transactions on Industrial Informatics, Vol. 1, No. 1, pp. 31-38.

Marik, V., Fletcher, M., Pechoucek, M. 2002a. Holons & Agents: Recent Developments and Mutual Impacts, In: V. Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.) Multi-Agent Systems and Applications II, Springer, Germany, pp. 233-267.

Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.). 2002b. Multi-Agent Systems and Applications II, Springer, Germany.

Marik, V., McFarlane, D. 2005. Industrial Adoption of Agent-Based Technologies, IEEE Intelligent Systems, Vol. 20, No. 1, pp. 27-35.

Maturana, F., Tichy, P., Staron, R., Slechta, P. 2002. Using Dynamically Created Decision-Making Organizations (Holarchies) to Plan, Commit and Execute Control Tasks in a Chilled Water System, Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA 2002).

Maturana, F., Tichy, P., Slechta, P., Staron, R., Discenzo, F., Hall, K., Marik, V. 2003. Cost-Based Dynamic Reconfiguration System for Evolving Holarchies, Proceedings of the 1st International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2003, Prague, Czech Republic, pp. 310-320.

Maturana, F. P., Staron, R.J, Hall, K. H. 2005a. Methodologies and Tools for Intelligent Agents in Distributed Control, IEEE Intelligent Systems, Vol. 20, No. 1, pp. 42-49.

Maturana, F., Staron, R., Tichy, P., Slechta, P. Vrba, P. 2005b. A Strategy to Implement and Validate Industrial Applications of Holonic Systems, Proceedings of the 2nd International Conference on Applications of Holonic and Multi-Agent Systems (HoloMAS 2005), Copenhagen, Denmark.

McFarlane, D., Bussmann, S. 2003. Holonic Manufacturing Control: Rationales, Developments, and Open Issues. In: Deen, S. (ed.) Agent-Based Manufacturing, Advances in the Holonic Approach, pp. 303-326.

Measarovic, M. D., Macko, D. & Takahara, Y. Theory of Hierarchical, Multilevel Systems, Academic Press, New York, USA, 1970.

Neligwa, T., Fletcher, M. 2003. An HMS operational model, In: Deen, S. M. (ed.) Agent Based Manufacturing: Advances in the Holonic Approach, Springer-Verlag, pp. 163-191.

Odell, J., Parunak, H. V. D., Bauer, B. 2001. Representing Agent Interaction Protocols in UML, In: In: Ciancarini, P., Wooldridge, M. (eds.) Agent-Oriented Software Engineering, Springer-Verlag.

Obitko, M., Marik, V. 2003. Adding OWL Semantics to Ontologies, Proceedings of the 1st International Conference on Applications of Holonic and Multi-Agent Systems (HoloMAS 2003), Prague, Czech Republic.

OPI. 2005. Odense Production Information homepage, http://www.opi.dk [referenced 31.10.2005]

Parunak, H. V. D. 1987. Manufacturing experience with the Contract Net. In: Huhns, M. N. (ed.), Distributed artificial intelligence, Pitman, London, UK, pp. 285–310.

Parunak, H. V. D. 1997. “Go to the Ant”: Engineering Principles from Natural Multi-Agent Systems, Annals of Operations Research, Vol. 75, pp. 69-101.

Parunak, H. V. D. 1999. Industrial and Practical Applications of DAI, In: Weiss, G. (ed.) Multiagent Systems. MIT Press, pp. 377-421.

Pechoucek, M., Vokrinek, J., Becvar, P. 2005. ExPlanTech: Multiagent Support for Manufacturing Decision Making, IEEE Intelligent Systems, Vol. 20, No. 1, pp. 67-74.

Pirttioja, T. 2002. Agent-Augmented Process Automation System, Master's Thesis, Helsinki University of Technology, Automation Technology Laboratory.

Pirttioja, T, Seilonen, I., Appelqvist, P., Halme, A., Koskinen, K. 2004. Agent-based Architecture for Information Handling in Process Automation, Proceedings of the 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS 2004), Vienna, Austria.

Rijnsdorp, J. E. 1991. Integrated Process Control and Automation. Process Measurement and Control, 2. Elsevier Science Publishers B.V.

Russell, S., Norvig, P. 1995. Artificial Intelligence, A Modern Approach. Prentice Hall.

Sanz, R. 2000. Agents for Complex Control Systems, In: Samad, T., Weyrauch, J. (eds.) Automation, Control and Complexity. Wiley & Sons, England, pp. 171-190.

Seilonen, I., Appelqvist, P., Halme, A., Koskinen, K. 2002a. Agent-Based Approach to Fault-Tolerance in Process Automation Systems, 3rd International Symposium on Robotics and Automation (ISRA 2002), Toluca, Mexico.

Seilonen, I., Appelqvist, P., Vainio, M., Halme, A., Koskinen, K. 2002b. A Concept of an Agent-Augmented Process Automation System, 17th IEEE International Symposium on Intelligent Control (ISIC 2002), Vancouver, Canada.

Seilonen, I., Koskinen, K., Pirttioja, T., Appelqvist, P., Halme, A. 2003a. Agent-Based Approach to Enhanced Flexibility in Process Automation Systems, Proceedings of the 3rd International Symposium on Open Control Systems 2003 (SoftSympo 2003), Helsinki, Finland.

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

Seilonen, I., Pirttioja, T., Appelqvist, P., Halme, A. Koskinen, K. 2003c. Cooperating Subprocess Agents in Process Automation, Proceedings of the 1st International Conference on Applications of Holonic and Multi-Agent Systems (HoloMAS 2003), Prague, Czech Republic.

Seilonen, I., Pirttioja, T., Appelqvist, P., Halme, A., Koskinen, K. 2004. Modelling Cooperative Control in Process Automation with Multi-Agent Systems, Proceedings of the 2nd IEEE International Conference on Industrial Informatics (INDIN 2004), Berlin, Germany.

Seilonen, I., Koskinen, K., Pirttioja, T., Appelqvist, P., Halme, A. 2005. Reactive and Deliberative Control and Cooperation in Multi-Agent System Based Process Automation, 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2005), Espoo, Finland.

Sen, S, Weiss, G. 1999. Learning in Multiagent Systems, In: Weiss, G. (ed.) Multiagent Systems. MIT Press, pp. 259-298.

Singh, M. P., Huhns, M. N. 2005. Service-Oriented Computing: Semantics, Processes, Agents. John Wiley & Sons, Ltd.

Singh, M. P., Rao, A. S., Georgeff, M. P. 1999. Formal Methods in DAI: Logic-Based Representation and Reasoning, In: Weiss, G. (ed.) Multiagent Systems. MIT Press, pp. 331-376.

Smar. 2002. DFI 302 – User's Manual, Smar International Corporation.

Tichy, P., Slechta, P., Maturana, F., Balasubramanian, S. 2002. Industrial MAS for planning and control. In: Marik, V., Stepankova, O., Krautwurmova, H., Luck, M. (eds.) Multi-Agent Systems and Applications II, Springer, Germany, pp. 280-295.

Tyan, C.-Y., Wang, P. P., Bahler, D. R. 1995. The Design of an Adaptive Multiple Agent Constraint-Based Controller for a Complex Hydraulic System, Proceedings of the International Joint Conference of CFSA/IFIS/SOFT'95 on Fuzzy Theory and Applications, Taipei, Taiwan, pp. 15-21.

Vaario, J., Ueda, K. 1996. Self-organization in manufacturing systems, Japan-USA Symposium on Flexible Automation, Vol. 2, ASME, Boston, pp. 1481-1484.

van Breemen, A. J. N. 2001. Agent-Based Multi-Controller Systems, Doctorate thesis, University of Twente.

van Breemen, A., de Vries, T. 2000. An Agent-Based Framework for Designing Multi-Controller Systems. Proceedings of the Fith International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, Manchester, UK, 10-12.4. pp. 219-235.

van Breemen, A. J. N., de Vries, T. J. A. 2001. Desing and Implementation of a Room Thermostat Using an Agent-Based Approach, Control Engineering Practice, Vol. 9, pp. 233-248.

van Breemen, A. J. N., de Vries, T. J. A., Striper, J. B. 2000. An Agent-Based Framework for Local Model Approaches. 16th IMACS World Congress.

Walker, S. S., Brennan, R. W., Norrie, D. H. 2005. Holonic Job Shop Scheduling Using a Multiagent System, IEEE Intelligent Systems, Vol. 20, No. 1, pp. 50-57.

Wang, H., Wang, C. 1997. Intelligent Agents in the Nuclear Industry, IEEE Computer, Vol. 30, No. 11, pp. 28-34.

Weiss, G. (ed.) 1999. Multiagent Systems. MIT Press.

Weiss, G. 2002. Agent Orientation in Software Engineering. Knowledge Engineering Review, Vol. 16, No. 4, pp. 349-373.

Wooldridge, M. 1999. Intelligent Agents, In: Weiss, G. (ed.) Multiagent Systems. MIT Press, pp. 27-77.

Wooldridge, M., Ciancarini, P. 2001. Agent-Oriented Software Engineering: The state of the Art, In: Ciancarini, P., Wooldridge, M. (eds.) Agent-Oriented Software Engineering, Springer-Verlag.

Wörn, H., Längle T., Albert M. 2002. Multi-Agent Architecture for Monitoring and Diagnosing Complex Systems, Proceedings of the 4th International Workshop on Computer Science and Information Technologies.

Wyns, J. 1999. Reference Architecture for Holonic Manufacturing Systems – The Key to Support Reconfiguration and Evolution, Doctorate thesis, Catholic University of Leuven.

Ygge, F. 1998. Market-Oriented Programming and its Application to Power Load Management, Doctorate thesis, Lund University, Department of Computer Science, Sweden, 224 p.

Ygge, F., 1999. Akkermans, H. Decentralised Markets versus Central Control: A Comparative Study, Journal of Artificial Intelligence Research, Vol. 11, pp. 301-333.

Yokoo, M., Ishida, T. 1999. Search Algorithms for Agents, In: Weiss, G. (ed.) Multiagent Systems. MIT Press, pp. 165-199.

HELSINKI UNIVERSITY OF TECHNOLOGY
INFORMATION AND COMPUTER SYSTEMS IN AUTOMATION

Report 1 Koskinen, K., Aarnio, P. (eds.),
Internet-, Intranet- and Multimedia Applications in Automation. June 1998.

Report 2 Koskinen, K., Aarnio, P. (eds.),
PC-based Automation Systems and Applications. June 1999.

Report 3 Mattila, M.,
Prosessilaitteen etätukijärjestelmä – ohjelmistoarkitehtuuri ja ohjelmistotekniset ratkaisut. March 2000.

Report 4 Strömman, M.,
Ohjelmoitavan logiikan ohjelmointi ohjelmistotuotantoprosessina. March 2002.

Report 5 Aarnio, P.,
Simulation of a Hybrid Locomotion Robot Vehicle. June 2002.

Report 6 Peltola, J.,
Uudet automaatiojärjestelmät - komponenttipohjaisen automaatiosovelluksen suoritusympäristö. September 2002.

Report 7 Fortu, T.,
Enterprise Resource Planning - Integration with Automation Systems. September 2002.

Report 8 Mattila, M.,
Condition Monitoring of an X-ray Analyzer. February 2003.

Report 9 Sierla, S.,
Middleware Solutions for Automation Applications - Case RTPS. June 2003.

Report 10 Honkanen. T.,
Modelling Industrial Maintenance Systems and the Effects of Automatic Condition Monitoring, February 2004.

Report 11 Seilonen, I.,
An Extended Process Automation System: An Approach based on a Multi-Agent System, February 2006.

ISBN 951-22-7976-2

ISBN 951-22-7977-0 (PDF)

ISSN 1456-0887

Picaset Oy, Helsinki 2006