Helsinki University of Technology Automation Technology
Series A: Research Reports No. 32
Espoo, December 2008

APPLYING AGENT TECHNOLOGY TO CONSTRUCTING FLEXIBLE MONITORING SYSTEMS IN PROCESS AUTOMATION

Teppo Pirttioja

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

Helsinki University of Technology Automation Technology
Series A: Research Reports No. 32
Espoo, December 2008

APPLYING AGENT TECHNOLOGY TO CONSTRUCTING FLEXIBLE MONITORING SYSTEMS IN PROCESS AUTOMATION

Teppo Pirttioja

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Electronics, Communications and Automation, for public examination and debate in Auditorium AS2 at Helsinki University of Technology (Espoo, Finland) on the 12th of December, 2008, at 12 noon.

Helsinki University of Technology
Faculty of Electronics, Communications and Automation
Department of Automation and Systems Technology

Distribution:
Helsinki University of Technology
Department of Automation and Systems Technology
P.O. Box 5500
FIN-02015 TKK
FINLAND

e-mail teppo.pirttioja@tkk.fi
Tel +358-9-451 5564
Fax +358-9-451 3308

ISBN 978-951-22-9685-9 (print)
ISBN 978-951-22-9686-6 (pdf)
ISSN 0783-5477

Picaset Oy
Helsinki 2008

Available on net at http://lib.tkk.fi/Diss/2008/isbn9789512296866

ABSTRACT OF DOCTORAL DISSERTATION	HELSINKI UNIVERSITY OF TECHNOLOGY P.O. BOX 1000, FI-02015 TKK http://www.tkk.fi
Author Teppo Pirttioja	
Name of the dissertation Applying Agent Technology to Constructing Flexible Monitoring Systems in Process Automation	
Manuscript submitted 12.6.2008	Manuscript revised 28.10.2008
Date of the defence 12.12.2008	
☒ Monograph	☐ Article dissertation (summary + original articles)
Faculty Faculty of Electronics, Communications and Automation Department Department of Automation and Systems Technology Field of research Automation technology Opponent(s) Professor Seppo Kuikka, Adjunct Professor Tapio Heikkilä Supervisor Professor Aarne Halme Instructor Adjunct Professor Pekka Appelqvist	

Abstract

The dissertation studies the application of agent technology to process automation monitoring and other domain specific functions. Motivation for the research work derives from the development of industrial production and process automation, and thereby the work load of operating personnel in charge of these large-scale processes has become more complex and difficult to handle. At the same time, the information technology infrastructure in process automation domain has developed ready to accept and utilise novel software engineering solutions. Agent technology is a new programming paradigm which has attractive properties like autonomy, flexibility and a possibility to distribute functions. In addition, agent technology offers a systematic methodology for designing goal based operations. This enables parts of the monitoring tasks to be delegated to the system.

In this research, new agent system architecture is introduced. The architecture specifies a structure that enables the use of agents in the process monitoring domain. In addition, an introductory internal layered design of an agent aiming to combine Semantic Web and agent technologies is presented. The developed agent architecture is used in conjunction with the systematic agent design methodology to construct and implement four test cases. Each case has industrially motivated interest and illustrates various aspects of monitoring functionalities. These tests provide evidence that by utilising agent technology it is possible to develop new monitoring features for process operators, otherwise infeasible as such within current process automation systems.

As a result of the research work, it can be stated that agent technology is a suited methodology to realise monitoring functionalities in process automation. It is also shown, that by applying solutions gained from the agent technology research, it is possible to define an architecture that enables to utilise the properties offered by agents in process automation environment. The proposed agent architecture supports features that are of generic interest in monitoring tasks. The developed architecture and research findings provide ground to import novel software engineering solutions to process automation monitoring.

Keywords agent technology, process automation, process monitoring, software architecture	
ISBN (printed) 978-951-22-9685-9	ISSN (printed) 0783-5477
ISBN (pdf) 978-951-22-9686-6	ISSN (pdf)
Language English	Number of pages 120 p.
Publisher Helsinki University of Technology, Department of Automation and Systems Technology	
Print distribution Helsinki University of Technology, Department of Automation and Systems Technology	
☒ The dissertation can be read at http://lib.tkk.fi/Diss/2008/isbn9789512296866	

VÄITÖSKIRJAN TIIVISTELMÄ	TEKNILLINEN KORKEAKOULU PL 1000, 02015 TKK http://www.tkk.fi
Tekijä Teppo Pirttioja	
Väitöskirjan nimi Agenttiteknologian soveltaminen joustavien monitorointijärjestelmien toteuttamiseen prosessiautomaatiossa	
Käsikirjoituksen päivämäärä 12.6.2008	Korjatun käsikirjoituksen päivämäärä 28.10.2008
Väitöstilaisuuden ajankohta 12.12.2008	
☒ Monografia	☐ Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)
Tiedekunta Elektroniikan, tietoliikenteen ja automaation tiedekunta Laitos Automaatio- ja systeemitekniikan laitos Tutkimusala Automaatiotekniikka Vastaväittäjä(t) Professori Seppo Kuikka, Dosentti Tapio Heikkilä Työn valvoja Professori Aarne Halme Työn ohjaaja Dosentti Pekka Appelqvist	

Tiivistelmä

Tässä väitöskirjassa tutkitaan agenttiteknologian sovelluskelpoisuutta prosessiautomaation monitorointisovelluksiin sekä niihin liittyviin toimintoihin. Työn motivaationa on teollisen tuotannon ja prosessiautomaation kehitys, jonka seurauksena prosesseja valvovalla henkilökunnalla on vaikeammin hallittavia yhä laajempia ja monimutkaisempia kokonaisuuksia vastuullaan. Toisaalta samaan aikaan prosessiautomaatio on kehittynyt tietoteknisenä ympäristönä mahdollistaen uusien tietoteknisten menetelmien hyödyntämisen. Agenttiteknologia on uusi ohjelmointiparadigma, jonka tärkeimpiä ominaisuuksia ovat autonomia, joustavuus sekä mahdollisuus toimintojen hajautukseen. Lisäksi agenttiteknologia tarjoaa systemaattisen tavan suunnitella tavoitepohjaista toimintaa, joka mahdollistaa monitorointitoimintojen osittaisen delegoinnin järjestelmälle.

Työssä esitellään uusi järjestelmäarkkitehtuuri, joka mahdollistaa agenttien hyödyntämisen prosessiautomaatioympäristössä. Lisäksi työssä esitellään myös alustava agenttien kerroksittainen sisärakenne, joka pyrkii yhdistämään Semantic Web - ja agenttitekniikat. Kehitettyä arkkitehtuuria on käytetty yhdessä systemaattisen agenttisuunnittelumenetelmän kanssa neljän teollisen motivaation omaavan esimerkkitoiminnallisuuden suunnitteluun ja toteutukseen. Nämä erilaisia monitorointitoimintoja esittelevät testit osoittavat, että agenttiteknologiaa hyödyntäen voidaan prosessioperaattoreille kehittää ominaisuuksia, joita nykyisillä prosessiautomaatiojärjestelmillä ei sellaisenaan ole mahdollista toteuttaa.

Tehdyn tutkimuksen pohjalta voidaan todeta agenttiteknologian soveltuvan prosessiautomaation monitorointitoiminnallisuuksien toteuttamiseen. Työssä osoitetaan, että soveltaen agenttitutkimuksen tarjoamia ratkaisuja on mahdollista kehittää arkkitehtuuri, jonka avulla agenttien ominaisuudet ovat hyödynnettävissä prosessiautomaatioympäristössä. Esitetyn arkkitehtuurin avulla voidaan myös saavuttaa ominaisuuksia joita monitorointitoiminnoilta yleisesti toivotaan. Kehitetty arkkitehtuuri ja tutkimustulokset luovat pohjaa uusien tietoteknisten ratkaisuiden tuomiselle prosessiautomaation monitorointisovellusten pariin.

Asiasanat agenttiteknologia, prosessiautomaatio, prosessien monitorointi, ohjelmistoarkkitehtuuri	
ISBN (painettu) 978-951-22-9685-9	ISSN (painettu) 0783-5477
ISBN (pdf) 978-951-22-9686-6	ISSN (pdf)
Kieli Englanti	Sivumäärä 120 s.
Julkaisija Teknillinen korkeakoulu, Automaatio- ja systeemitekniikan laitos	
Painetun väitöskirjan jakelu Teknillinen korkeakoulu, Automaatio- ja systeemitekniikan laitos	
☒ Luettavissa verkossa osoitteessa http://lib.tkk.fi/Diss/2008/isbn9789512296866	

Preface

Many past hobbies and people have had an influence on my ideas of how things should and could be working within industrial settings. Concretely the story behind this thesis began in year 2001 as I became to work on my masters thesis in the Automation Technology Laboratory of Helsinki University of Technology, Finland. Since then I have had many days when the original enthusiastic beliefs reaching towards unrealistic goals have not been enough and completion of this dissertation could not have succeeded without the help of several key people.

First of all, I would like to thank my supervisor, Professor Aarne Halme, for his support and guidance in my work. Especially I'm grateful to him as the head of the laboratory giving me 100 % support for my personally motivated research subject and the possibility to pursue the research for long enough.

I thank my instructor Adjunct Professor Pekka Appelqvist for personal guidance both in research issues as well as in a variety of practical matters.

This thesis is the result of research work carried out during the period 2002–2008, mainly in research projects funded by the Finnish Funding Agency for Technology and Innovation (TEKES). These projects had a varying number of personnel, but in addition to the above mentioned three key persons have influenced the subjects reported in this thesis more than others. I thank Dr. Ilkka Seilonen for promoting structural thinking relatively often and also for being available for academic discussions about the research subject. Further, I would like to thank Mr. Antti Pakonen from VTT for bringing positive and forward pushing attitude to the research group. It did make a difference. In addition, without support from Professor Kari Koskinen the research group would not have had the possibility to work with agents as long as it did.

Furthermore, I would like to thank my preliminary examiners Professor Seppo Kuikka and Dr. Olli Ventä for their reviews and valuable comments on the thesis.

Thanks are also due to the lab personnel for providing an inspiring atmosphere and especially the people around the table hockey that made it easier to relax when things started to be too much. Additionally, it always gives new ideas when you see innovative gadgets moving around the lab's corridor.

I am very grateful for the financial support from Emil Aaltosen Säätiö and Tekniikan Edistämissäätiö. Additionally, I would like to thank Johanna Mitjonen for giving me the opportunity to unload some of the issues about the thesis work as a columnist for the Polyteekkari magasine.

I would like to thank the supportive people outside the academic world. Firstly, I must thank all my friends for constantly promoting healthy sports and other fun stuff. Further, hugs and kisses to mom, dad, and my little sister.

Finally, I owe thanks to Nina Virtanen for bringing heaps of happiness to my life.

Helsinki, November 2008

Teppo Pirttioja

Table of Contents

List of Abbreviations

AI	Artificial Intelligence
AOSE	Agent-Oriented Software Engineering
BDI	Belief-Desire-Intention (model)
CP	Change Point
CSP	Constraint Satisfaction Problem
DCS	Distributed Control System
EDDL	Electronic Device Description Language
ERP	Enterprise Resource Planning
FDI	Fault Detection and Identification
FDT	Field Device Tool
FIPA	Foundation of Intelligent Physical Agents
IOPE	Input, Output, Pre-condition, and Effect
IT	Information Technology
KPI	Key Performance Indicators
LIMS	Laboratory Information Management System
MAS	Multi-Agent System
MES	Manufacturing Execution System
OEE	Overall Equipment Effectiveness
OPC	Open connectivity via open standards
OPC UA	OPC Unified Architecture
OWL	Web Ontology Language
PID	Proportional Integral Derivative (Controller)
PLC	Programmable Logic Controller
RDF	Resource Description Framework
SCADA	Supervisory Control And Data Acquisition
SOA	Service-Oriented Architecture
SPARQL	Simple Protocol and RDF Query Language
SQL	Structured Query Language
SWS	Semantic Web Services
XML	Extensible Markup Language
W3C	World Wide Web Consortium

1 Introduction

1.1 Motivation and background

Technological changes originating both from the development of the process automation environment itself and from business requirements are altering the way in which process industry-related systems should operate in the future. New technical solutions have been utilised to stay competitive in this change situation. This research also falls into this category; it studies how agent technology, as a new software engineering paradigm, could be used to build better monitoring systems in the future.

Digitalisation and advances in embedded electronics are currently contributing strongly to the evolutionary and continuous development of automation environments. Sophisticated field devices provide diagnostic services and wireless communication may be used to distribute additional but beneficial information everywhere in the system, and this may be realised even without interfering with the control operations. General information technology-based (IT) solutions developed initially for office environments have become so versatile and mature that many of the utilities provided have also been adopted for process automation. This has added applications such as electronic manuals and reporting systems to the daily life of process operators. In addition, tighter integration into enterprise-level systems is presently under development, e.g. with Manufacturing Execution Systems.

The maturity of process automation has provided a chance to increase the size and complexity of the systems, and, for example, only a few operators are needed to be able to run a chemical production site with thousands of I/Os. Normally, when there are no abnormalities or changes and practically nothing happens this minimal setup is realistic. But when there is something out of the ordinary, e.g. a device breaks down or there is a blockage in the pipes, then a few persons watching over changing values from hundreds of displays is not enough. Models describing the relations of quantities and limits of process operations have offered a solution to overcome this problem, and with unit processes this approach has been demonstrated to work relatively well. However, when a larger part of the system, with numerous changing interconnections, has to be monitored, the model-based approach becomes unmanageable.

Operator work in process sites has become more and more like regular knowledge-intensive work. Information is searched for and processed extensively with various IT systems. However, IT solutions originally developed for office environments are not, as such, suitable for all the tasks in process environments, especially not for monitoring. Contrary to office environments, where the modifications to stored information are mostly made by humans, the significant changes to information in process environments are a result of unknown phenomena and uncontrollable variables. In processes the changes are a consequence of e.g. device malfunction, variations in raw material, and the unknown cross-effects of different process sections. These changes may not be *a priori* controlled by set practices, e.g. the ISO 9001 quality management standard, as may be done with business-related systems in office environments.

The demands of economic competitiveness and environmental issues require an increase in production effectiveness, which is typically responded to with more integrated production systems. As the amount of buffer storage has decreased and the time from order to production has been minimised at the same time, the whole production system has become more vulnerable to malfunctions. Furthermore, the demand for increased overall flexibility and the ability to change a production setup relatively fast and without material losses has increased. All these changes alter the controlling setup that process automation realises. Changes in control systems also reflect how monitoring is conducted.

Highly developed IT has provided the possibility of offering new types of services to people in their everyday lives and, especially, the monitoring of information sources is starting to be commonly available. Nowadays a notification request may be left with an online bookshop if a certain book is not currently ready to be ordered. Similar services are also available in banking, for example an alert about your balance going below a user-specified limit. With these services the user may delegate the monitoring activity by defining triggering values and then wait to be informed by the service provider. A variety of monitoring services about interesting information is also becoming available, for example everywhere on the internet (e.g. google.com/alerts) or via various online news services. The availability of these services is making people familiar with this type of operation and, furthermore, makes it easier to require similar functions to be available within process automation environments.

The development of general network technology and, especially, the internet has highlighted the importance of software engineering tools for decentralised and distributed systems. On the one hand, a growing user community is continuously developing new applications utilising the possibilities that the network provides. The results are visible, for example in peer-to-peer networks that operate without central control, thus offering the potential for enormous adaptation in terms of scale and availability. Social computing, where the users provide and develop the content, is the new buzzword in the software area. On the other hand, industry has been keen to utilise the network-related possibilities for its own purposes. Web services have been developed for organising commercial tasks between players in the production chain, and Service-Oriented Architecture is currently making a strong forward push in global organisations.

In the academic world, agent technology originating from the artificial intelligence community has a long research tradition of organising operations in distributed settings. Furthermore, agent technology has been proposed as a suitable realisation tool for reducing work and the information overload on humans via delegation. The Semantic Web has been demonstrated to ease the solving of interoperability problems in applications that integrate multiple individual actors and link pieces of information together dynamically. The Semantic Web is still under heavy development but even now solutions for systematically organising knowledge representation issues are relatively mature, both in terms of actual tools and also standards.

The results presented in this thesis are the outcome of multiple projects studying the possibilities of agent technology within process automation. The projects were open-minded from the very beginning and were based on an experimental approach. The first of these projects was *Agent-based automation systems*, conducted in 2000-2003. It was focused on finding new ways to structure the operation of an automation system utilising agent technology. The ideas at the beginning were radical, including e.g. 3D temperature profiles measured by mobile sensors (Appelqvist et al. 2002) and fault recovery with emergent behaviour. However, the ideas were found not to be feasible by the research group when they were studying the properties of a process automation environment and its functions. The author joined the research group in the year 2001 to do his master's thesis, which was finalised in 2002 with the title "Agent augmented process automation system" (Pirttioja 2002). By that time, the research group was mainly studying the structural aspects of agent systems and their applicability to the controlling of operations. Additionally, the project highlighted the possibility of utilising agents in information-processing functionalities.

The second project, *Adaptive automation* (*MUKAUTUVA)* in the years 2003-2004, turned the research focus more towards information processing and demonstrated simple information-accessing tasks within process automation setups. The third and the last project so far was *Agent-based information services for process automation* (PROAGE), carried out during the years 2005-2007. This was totally focused on information processing and by then the Semantic Web had been included into the architecture because of the general developments originating from the World Wide Web Consortium (W3C). The results in this thesis are mainly derived from the final project, although the previous two projects provided important background information about both the problem domain and the agent technology itself.

Although this thesis reports the results on the information-processing functions, it is strongly influenced by the previous research done by the research group on applying agent-based approaches to the supervisory control operations of process automation. That research presented a specification of how an agent system could be used as an extension to an existing automation system and thus make possible the use of agent technology to provide enhanced reconfigurability, responsiveness, and flexibility to process automation control operations. The main results from these issues can be found in Seilonen's thesis (2006). Although the general background of these two research subjects is somewhat similar, especially using agent-based systems as a skeleton to gain flexibility and reconfigurability, there are still major differences. In control functions the overall goal is always to make some kind of modifications to the system setup, more or less automatically. In monitoring, which is a specific type of information processing performed in a process automation environment, the purpose of the designed system is to provide the user with flexible, integrated, and easier access to available information. These types of functions (e.g. monitoring and diagnostic) seemed to be more suitable for agent orientation than the controlling functions, and similar statements have also been introduced by other researchers (Bunch et al. 2004; Marik and McFarlane 2005; Wagner 2002).

1.2 Research problem and goals for the study

The research problem in this thesis is to draw conclusions on usefulness of agent technology when used to build a monitoring system for operators working in a process automation environment. To be able to answer this, the requirements of such a system have to be determined and then the properties of an agent system designed and built for the purpose should be analysed.

As the size and complexity of production sites is increasing, it will become generally problematic to find out if and when certain information is available, and, further, to decide if it is relevant in the current situation. Reducing the mental load of an operator is crucial and the demands for technical solutions assisting people in their work are high. The technical background for developing advanced services is relatively mature but it is rather unclear what properties and functions are required from the resulting technical construction. Applications meeting individual requirements and building on top of present structures have been presented, but the current risk-minimising research culture seems to fail in providing the badly-needed ground-breaking results. Open-minded research demonstrating the possibilities of structurally new solutions has only been presented to a very modest degree, and this also justifies the study of the potential of agent technology.

The research problem of the thesis can be defined with a general research goal as follows:

Develop a system design and provide a methodology that, with the help of novel information technologies, answers the current and future monitoring needs in process automation. Release the human user from the need to perform regular checking of the current process state and provide controllable and configurable timely access to filtered, processed, abstracted, and adapted information available from heterogeneous data sources based on user-defined conditions referring to the values themselves and their relations. This system should provide the means for easy and comprehensive monitoring of process operations and related information, even when it is incomplete and its availability is possibly partly unknown.

Further, this general research goal can be divided into the following more concrete objectives:

- *Study agent technology opportunities and their suitability for monitoring functions:* changes are visible in process automation environments, originating both from technological development and business trends. How these are altering the everyday work of personnel working with processes are unclear, but, on the basis of the literature, some predictions may be made. Because of its proposed general properties, agent technology seems to be suitable but various alternative operating principles need to be compared and the most suitable ones need to be selected. Furthermore, other state-of-the-art technological alternatives need to be studied.
- *Specify agent architecture for monitoring applications:* a list of the properties that the system should have needs to be gathered on the basis of a background study. The system architecture should be usable in a real automation environment, utilise the latest technological opportunities, and further support the realisation of new functionalities. Utilising agent technology, a number of

concrete design issues need to be decided before the whole system structure is ready for realisation. The system should be specified in enough detail for it to be implementable.

- *Construct industrially motivated monitoring functionalities with agents:* use the agent philosophy and the developed architectural design to construct selected monitoring functionalities. The realisation of the functionalities of experiments enables the agent methodology and, further, how the desired system properties are met to be assessed. Furthermore, the experiment provides the possibility of suggesting future improvements in the design.
- *Critical evaluation of the research:* Provide a concluding discussion about the whole area of agent technology-based monitoring system design. Discuss the presented monitoring system requirements and their relevance. Summarise how the architecture that was designed and system that was constructed performed and consider whether the arguments presented are well justified. The performance of the experiment provides the possibility of making a critical assessment about the properties that the agent technology and the designed system do and do not provide. Discuss also the potential limitations of the technology.

Although the thesis is about designing a system for process automation environments, hard real-time issues were decided not to be considered. The reason for this is that it is seen that higher-level monitoring information is useful, even if it takes some seconds to have it available.

The research work in this thesis is done on a holistic level; it aims to facilitate an increase in knowledge about the whole technological construction and thus does not focus on any individual part of the system. The research is divided into two distinct areas of theoretical and experimental work which are tightly connected in an interactive and iterative fashion. The related research was studied in order to get an understanding of the challenges, requirements, and available technical opportunities in the application area. The experimental work, using a constructive research methodology, was carried out in order to test the properties and benefits that selected approaches were suggested by the literature as having. In addition, experiments were done to demonstrate the applicability of the chosen technologies and to gain an understanding of the approaches used and design choices selected. The Prometheus agent design methodology (Padgham and Winikoff 2004) was used in the experiments because at the time of the research it offered the best support for practical usage.

1.3 Contributions

The contributions of the thesis include the following:

- *Agent technology opportunities compared to process monitoring requirements:* previous research has indicated that agent technology is suitable for similar operations to those requested in monitoring tasks. However, this research compares the presented agent technology properties to new monitoring requirements originating from business changes within process industry. In particular, the flexibility requirement is naturally supported by agent-oriented design.

- *Specification of agent architecture for monitoring in process automation, including the BDI model and the Semantic Web:* the desired properties for a monitoring system situated in process industry are gathered. On the basis of these an agent system architecture is designed, covering the general system structure and the roles of the agents. The architectural design utilises the latest available solutions for goal-oriented operation (with the Belief-Desire-Intention model) and knowledge presentation technologies, originating from Semantic Web research. Further, a layered and modular internal structure of an agent supporting the hierarchical organisation of data-processing in the process automation context is specified. In addition, an actual example of software realisation is presented.
- *Application of systematic AOSE method to design industrial monitoring cases:* the design of four industrially motivated experiments is documented. These experiments illustrate the use of systematic AOSE in the construction of monitoring functions in a process automation environment. The specified agent system architecture is used to implement the experiments.
- *Evaluation:* the applicability of an agent approach is critically evaluated on the basis of a literature review, synthesis, design, and experiments. This provides insights into what the suitable monitoring functionalities are for a system that has been built with an agent approach. It also discusses what other information technologies should be used when developing this kind of monitoring system.

1.4 Author's contribution within the research group

The research work documented and presented in this thesis was carried out from 2002 until 2007 in a group that had three key members. The author's main scientific contribution to the group has been the architectural design. The desired properties for the agent system presented in the thesis and the specification of information related roles (client, information, and wrapper) of agents were done by the author. Furthermore, the internal layered structure of an agent was specified by the author. The use of constraint satisfaction problems formalism to monitoring purposes was proposed and implemented by Ilkka Seilonen. In addition, the presented process industry data model and ontology based information processing were mainly developed by Antti Pakonen.

Experiments 1 and 2 were defined, designed and implemented by Pakonen and Seilonen in co-operation with the author, but the agent design presented in the thesis with Prometheus formalism was done solely by the author. Furthermore, specifications and design for Experiments 3 and 4 were made exclusively by the author. Furthermore, the author is alone responsible for the idea of applying statistical mathematical tools to find linear episodes from process time series data. The implementation was done by Eemeli Aro, based on author's definitions.

1.5 Outline of the thesis

This thesis is organised as follows:

Chapter 1: Introduction.

Chapter 2: *State of the art of monitoring systems in process automation*. The chapter presents the current status of industrial process automation and its monitoring. It describes the general motivation for the research and presents the user needs.

Chapter 3: *Agent, semantic web technologies, and their application to monitoring task in process automation.* The research depends heavily on information technology, so the available system-level solutions are reviewed. In addition, a synthesis discussing the challenges and technical opportunities of process automation monitoring is presented.

Chapter 4: *The new agent system architecture for process monitoring*. First, the desired system properties are listed, and these properties direct the development of the architecture. Then the architectural design is presented, with a summarisation of its properties.

Chapter 5: *Layered agent design and its functionalities.* The layered and modularised design of an agent used further in the study is presented. The design is used to implement and realise the system for experiments.

Chapter 6: *Experiments illustrating the usefulness of agent design methods and developed architecture.* Four different real life-motivated experiments are designed and described in detail. Each experiment is introduced with its use case and then the agent definitions and interactions are designed. Furthermore, the implementation of the experiment and the concluded tests are illustrated.

Chapter 7: *Discussion and conclusions.* The results of the thesis as a whole are summarised on the basis of the previous chapters. This contains discussion about how the properties offered by the literature match the results from the architectural design and experiments. Future work is also pointed out.

2 State of the art of monitoring systems in process automation

2.1 Introduction

Process automation is a special branch of automation that is used to realise the automation of continuous industrial production processes, such as paper production or the chemical industry. This chapter presents the characteristics of process automation and its current state. It presents the visible future in terms of technical development trends and business requirements, and discusses how these are reflected in process monitoring. It also describes the current status of industrial automation, because the industrial practice and current technical setup are important as they represent the environment in which the developed monitoring system will be situated. The chapter will first present the automation environment from three distinct perspectives of time, technology, and users, in their own subchapters. Then, finally, it illustrates the overall development trends that are currently visible in process automation. These will all be used as background information when designing the system for monitoring purposes in later chapters.

Generally, the technical motivation for automation is that some systems are too fast or complicated to let humans control them directly, or machines handling the repetitive tasks let people focus on higher-level aspects of work, e.g. optimisation and quality control. The level of automation varies depending on a number of issues, such as safety requirements and business interests (Olsson 1992; Sheridan 1992). Because of partly distinct properties and functionalities, process automation in continuous processes has differences in its technical requirements compared to other domains, e.g. manufacturing.

Automation that controls processes varies in type and in its place in the life cycle, which in process automation may be as long as 20 to 30 years. Typically, the systems controlling the operation of production plants are based on mixed technologies and product life cycle statuses vary. As most plant systems work properly in their operational life cycle, some parts of the system are at the end of their life cycle. For some parts requirements are gathered for the next system upgrade, and possibly some previously updated part is starting to meet the designed production state. As the system goes through continuous evolution the whole life cycle of systems is important and this should be remembered when developing new functions. This evolution makes the construction of the automatic monitoring of processes especially difficult, as rules expressing acceptable process values and operational states may become out of date in a relatively short time, e.g. because of the wearing out of equipment or changes in the outside temperature.

Although the whole evolutionary path, including design and commissioning, is important, this thesis focuses on studying issues related to the normal operational phase. This phase is nominal for the process system and its correct operation is the responsibility of the operator. According to Paunonen (1997), working with process automation in this operational phase may be divided into the following modes:

- *Monitoring* of the normal operation of the process when it is steady and it is working as planned. This could be seen as the target state of the production plant.
- *Performing preplanned tasks* is a mode that is activated once in a while and it may also be seen as the target state of production. This consists of situations such as process start-up sequences and grade changes.
- *Controlling disturbances* is a situation in which things are not going as planned, e.g. when some equipment has broken down and the production capabilities of the site are reduced. Before this phase is activated, the deviation from normal operation (the previous two phases) needs to be observed.
- *Development* mode is activated when the on-going evolution of the production system requires modifications to the system setup.

Each of these modes requires a different perspective on the underlying system and its operation, and therefore the tools that are designed to support user activities in different modes should take this issue into account.

2.2 Evolution of process automation

The development path of process automation and its technologies has been rather isolated in the past, and although this isolation is effectively in a state of breaking up, the history still affects the trends and development directions. Therefore, a brief overview of the evolution of the technological background and the growth of the general size and complexity of applications gives a perspective that may be exploited when new systems and functionalities are being developed.

The first control systems were based on mechanical operation principles. In general, mechanical control systems work quite well when done properly, but they wear out with use, need extensive maintenance, and are hard to reconfigure. Although mechanical systems are not thought to be a feasible implementation of modern control functions, their monitoring was relatively easy because of their simple and intuitive operation. From purely mechanically constructed control setups the development went through electronic relays and hydraulic and pneumatic systems in the 1950s to digital control systems somewhere round the 1970s. With these the control systems were constructed mainly from multiple single-variable loops that handled their own share of the whole system. Operations were easy to validate and monitor directly with human perceptions.

More advanced control systems became available round the 1980s as computational systems developed further. At first, automation-related hardware and software were specially designed for control applications but more recently the hardware used has become more and more similar to that found in office computers. With totally all-digital platforms a number of new functions became possible, such as plant-wide optimisation. Additionally, the alarm functionality provided the possibility of requesting the system to give notification about individual values going over specified limits. Digitalisation made it possible easily to move a larger part of the process so that it could be supervised from one central control room. This made possible the use of fewer persons to watch over the larger part of the process.

In the 1990s a wide range of digital field buses were introduced to process industry. Field buses provided error-free communication between controllers and field devices, and made it possible to deliver additional information between devices. Along with the development of microcontrollers, this made room for embedded diagnostics, and maintenance tools started to emerge. In addition, around the 1990s control rooms got bigger displays and navigation between different process layout displays became easier. At the same time, the division between control systems and information systems ceased to be so obvious any more and these started to merge (Paunonen 1997). As the fusion of information and control systems became technically feasible, the responsibilities of process operators again increased.

Around the year 2000 the development focus in process industry moved more and more towards the utilisation of the potential that the software discipline offered. The development of network technologies, e.g., the internet and Ethernet, adapted from offices also made new levels of integration possible within factories, and the integration of the whole production enterprise became more important. Despite the integration possibilities and intelligent machines, which, together, offer great possibilities, especially in terms of flexibility, systematic development methods for interoperable components were found to be largely lacking (Tommila et al. 2001). However, technically easier integration made various device maintenance, diagnostic, and monitoring tools really useful. But even today, a need for more effective system-wide integration, allowing systems developed in isolation to communicate, is still seen (Laukkanen 2008; Ventä 2005). For monitoring operators this development has offered easier navigation to external applications, for example links to device manuals and access to work orders. Lately, using single solution providers' integrated process automation systems has enabled the raw measurement data from the plant floor to be available at the right time, in the right form (ABB 2007; Metso 2007; Rockwell 2005, 2006, 2007). All these improvements have made it possible to increase the area that a single operator is responsible for.

Lately, the development drive has been towards the more tightly organised integration of information systems to support managers in their need for more accurate and timely information about the production situation in the whole environment (Jämsä-Jounela 2007; Laukkanen 2008; Ventä 2005). Business measurements, such as Overall Equipment Effectiveness (OEE) and other Key Performance Indicators (KPI), are also gaining interest in process industries. These integrate physical operational information with economic measures and require correct and timely data from all around the production site in order to be applicable (Tuomaala 2007). Because of the level of integration and effectiveness required, production systems as a whole are growing in size, complexity, and "accuracy", and the amount of data generated and stored is also growing vastly. For example, Laukkanen (2008) states that in paper production processes this trend has resulted in the integration of the problems and disturbances.

Lately, the importance of information issues in industry has gained notice, and as a result of this the *IEEE Transactions on Industrial Informatics* (TII 2008) journal was established in 2005. In addition, the IEEE International Conference on Industrial Informatics conference series has received growing attention since it began in the year 2003.

2.3 Current technological solutions

Present-day automation infrastructure is based on multi-level and hierarchical networks of diverse devices that communicate with each other using various methods. Although analogue communication is still widely used to connect field-level devices, it can be said that the majority of communication among enterprises is digital (Sauter 2005; Ventä 2005). This has provided a lot of integration possibilities among plant and enterprise networks. The exact number of individual hierarchical levels varies between different solutions but the general situation is illustrated in Figure 1.

On the lowest level of the whole automation infrastructure are field devices or instrumentation, which have benefited significantly from general electronic developments. Novel all-digital field devices have numerous added functionalities along with their basic operation, e.g. self-diagnosis (Jämsä-Jounela 2007). In addition, wireless and ubiquitous technologies are introducing new possibilities for distributed information generation, which are useful at least for diagnostic purposes (Elmusrati et al. 2007). However, there are no clear ideas or methodologies as to how all the information available could be fully utilised on other levels of production enterprises. The present practice is mainly based on manual operation, where maintenance personnel use standalone software products provided by each device vendor. The development of information models and communication standardisation for device interoperability is ongoing, e.g. EDDL (2008) and FDT (2008), but the requirement for more open-minded research has already been noted (FDI-future 2007).

Hardware has developed considerably since the first process stations and controllers were introduced. Earlier, much of the complex data processing, such as the heavy computational work of optimisation, was done in separate systems for practical reasons. Nowadays there is plenty of computational power available, and the statistical calculation used in the paper industry for quality assurance and process optimisation may be performed within process stations. Furthermore, process stations have become capable of supplying all the measured data from field devices to everywhere within the enterprise, if needed. The methodologies used to programme process stations have essentially stayed the same, focusing on supporting the configuring of control functions. Supporting functionalities, such as optimisation, diagnosis, monitoring, reporting, and so on are usually programmed and carried out separately. This makes maintaining these functionalities cumbersome and laborious, and the future research should address this weak point and develop more flexible structures.

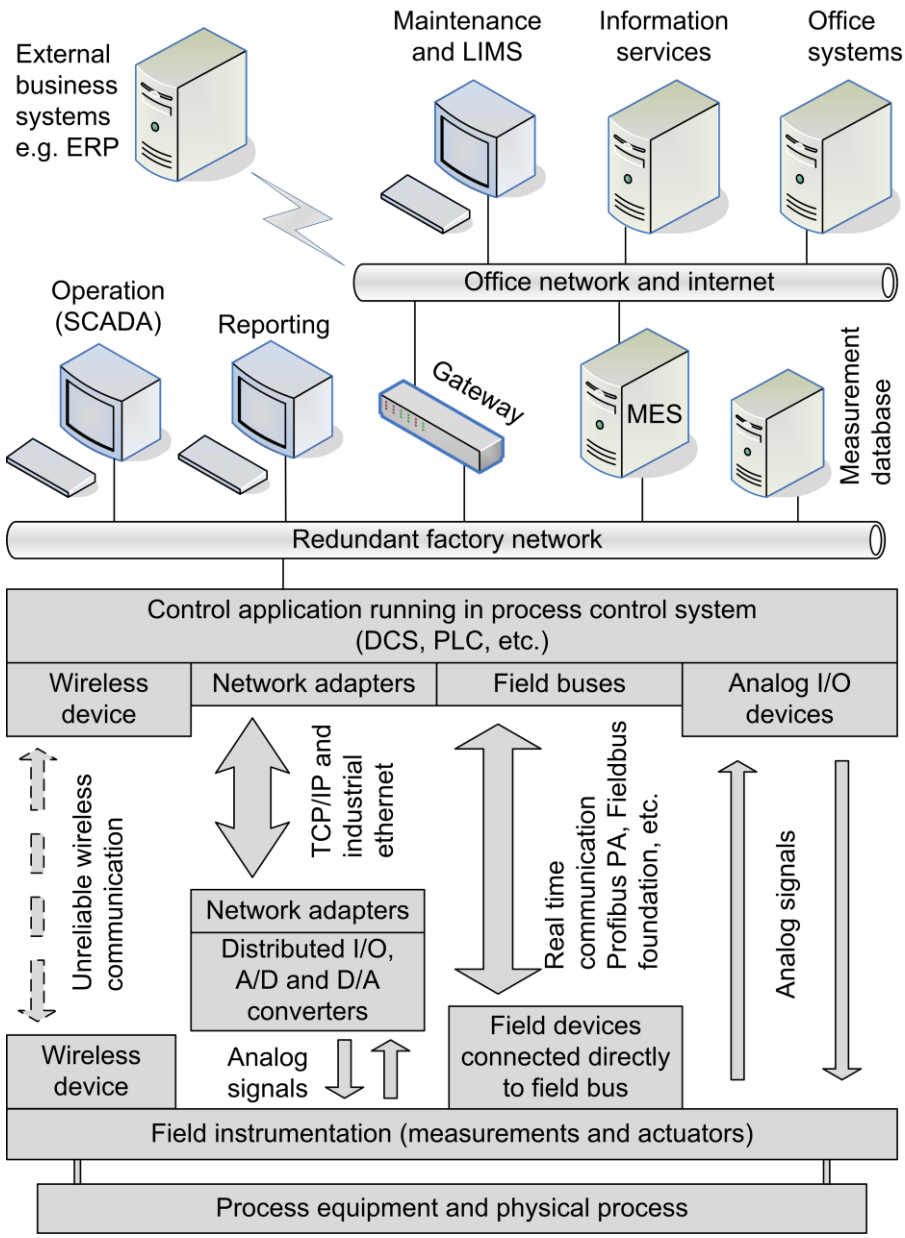

Figure 1 – Multiple layers of modern automation infrastructure.

Practically every automation system, platform, and solution has its own variant of how the control application is programmed. Unit controllers measuring and controlling one pair of values are only configured, e.g. a PID controller with its respective parameters. PLCs are programmed with rather simple languages, such as ladder logic. The control application in process stations is usually performed with function blocks, and a number of programming standards have been created for this. Actual implementations more or less follow these standards.

Function blocks are configurable blocks that implement some process control-related functionality, e.g. a PID controller. The whole process control functionality is set up by defining a network of these functionalities and configuring the parameters of each function block. Function blocks are like objects in Object-Oriented programming, with their own internal operating principles, protected variables, and well-defined connection interface to the outer world. The standard IEC 61804 Function Blocks for Process Control is one example of a language that implements this principle. In

addition to function blocks, logic languages are widely used in industry for control purposes. The most commonly used set of these languages is described by the standard IEC 61131-3 (Programmable controllers, part 3, Programming languages) developed for PLCs. For further information about this standard see Lewis (1998). These languages are heavily based on legacy ideas of how to construct control setups, e.g. the language named *Ladder logic* emulates the physical wiring of relays and switches.

Although the technical progress within automation devices and systems has been tremendous in the past couple of decades, control application programming with function blocks or logic languages has remained rather untouched. The present-day industry standard languages for control software development have been designed for realising rather simple control structures. A justifiable reason for this is that reliability, with compact and verifiable operation, was the main goal when developing these languages. Furthermore, with these languages the user was able to design those control functionalities that most effectively use the computing power that was limitedly available in the controlling hardware. This has resulted in languages that do not effectively support the handling of complex structures and functionalities. In particular, the available languages do not support the construction of flexible diagnostic or monitoring functionalities, and organising the advanced functionalities is practically left to the user.

Maybe the most important novel solution within automation programming is the IEC 61499 Function Block standard, originating from holonic manufacturing systems research. The standard aims to provide more open architecture in order to gain portability, interoperability, and configurability to industrial measurement and control applications (Deen 2003). This standard is more flexible than the previous ones and it allows the development and building of more advanced functionalities. But as the standard allows more complex structures to be built it makes the basic design and application development more vulnerable to errors and requires a complex platform which itself is also a challenge to reliability. The IEC 61499 standard is still under heavy development and no large-scale industrial reference cases have occurred yet, so no true comparison can be made. However, the event-based operating principle in IEC 61499 might generate new possibilities in the development of monitoring, diagnostics, and abnormal operation functionalities in general.

In the process automation environment more information will be gathered and the need for integration with other information sources will rise. However, because of the programming principles currently used on the control level the connection between these is rather laborious and cumbersome to organise. Currently, information integration is typically realised with a list of shared variables between systems and this requires time-consuming agreement between players to build a shared understanding of the rationales behind the variables and their values. Technically, integration between systems has been realised with the OPC Data Access interface, designed for exchanging real-time process information (OPC 2008). In OPC Data Access the interface specification is organised as hierarchical groups of quantities, the values of which the client may ask for. In the future a more intelligent solution for this integration procedure will definitely be needed (Ventä 2005).

When physical production, with its control system, is connected to a global enterprise the setup gets even more challenging. Technically, this integration between factory floor instrumentation, control systems, and office automation was demonstrated years ago, so this vertical integration is no longer a technical problem (Jämsä-Jounela 2007; Sauter 2005). The Manufacturing Execution System (MES) is a solution used for vertical integration in discrete manufacturing, and it has lately also gained interest in process industry. MES is used as an active data-processing link between control-level information and office systems, and its purpose is mainly to relay timely information between systems operating on different levels of abstraction (McCellan 1997). The forthcoming OPC Unified Architecture, which has a more service-oriented approach, offers interesting possibilities for the realisation of integration between all levels of automation (OPC UA 2008), but the standard does not yet specify the data models needed to realise this.

Currently, the integration of functionalities is typically based on static configuration, which is done at the time of design by the design engineer. Methodologies supporting dynamic integration in horizontal and vertical directions in changing situations are largely lacking and future research should address this (Ventä 2005). From the business perspective flexibility is becoming a more important property, also in the process industry (Jämsä-Jounela 2007; Keller and Bryan 2000) and new technical approaches are being researched, e.g. agent technology has been proposed (Chokshi and McFarlane 2008). Additionally, the need for temporal integration covering the whole life cycle of the system has been pointed out in research (Sauter 2005). Similar information integration issues, where entities from different application areas are logically related, are resolved with semantics in the electric power market (Huang and Lei 2007).

2.4 User perspective of process monitoring

In the future, more functions and operations will be able to be automated with more capable systems, but still at the same time the importance of the human supervisory role seems to stay (Gentil 2006; Sheridan 1992). Although controlling systems are generally developed for fully automatic operation, Paunonen (1997) has proposed that showing the information to the user is also one of the primary tasks of control systems within industry. Part of this information is exploited within monitoring purposes on multiple levels of production sites. As part of their daily business, operators are watching over real-time controllers and verifying that the process is working safely, correctly, and as efficiently as needed. This is called on-line monitoring. Their task is to be aware of the present process state and minimise possible production losses with detection of device malfunctions and other abnormal situations as early as possible. Maintenance personnel, on the other hand, are looking at things from a longer-term perspective, focused more on device malfunctions, and their work is to try to estimate the correct timing for maintenance. Furthermore, management personnel focus on much higher-level data that concern elements of the physical process less directly.

In the future operators will be more responsible for productivity from the business perspective, e.g. decisions at management level will cascade down to the automation system in real time (Laukkanen 2008). Simultaneously, as the work of the operator is broadening in terms of what aspects of the production are required to be monitored, the volume of monitored items will also increase. For example, it has been reported that the I/O number that one operator is responsible for has grown from approx. 400

in the year 1990 to 1200 in the year 2005 and will be as high as 2600 I/Os in the year 2006 (Laiho 2005).

In control rooms the interface development has so far been going mostly where technological developments have been taking it (Farbrot et al. 2000), and revolutionary new ideas have largely been lacking (Ventä 2005). To perform well in dynamic situations an operator needs timely information from the process itself and also from the all-around enterprise in order to be able to make good decisions (Laukkanen 2008; Ollson 1992; Paunonen 1997; Sauter 2005; Sheridan 1992; Ventä 2005). One of the most important monitoring tools for operators is alarms, but, at least currently, they are designed with a single process component in mind and typically using static limits (Farbrot et al. 2000), which makes them practically useless in situations where the process setup changes dynamically. Although the process operators are responsible mainly for the on-line monitoring, Paunonen suggested as early as in the year 1997 that some parts of the off-line monitoring tasks, e.g. searching for malfunctions from historical information, will also be added to operators' tasks (Paunonen 1997).

As sites get bigger and there is more to keep an eye on for every process operator, there is a need to develop technology that helps detect various abnormal events faster (Laukkanen 2008) and at the same time keeps the mental load of the user on an acceptable level (Paunonen 1997). These functions should also depend on the current situation and status of the process (Ventä 2005). Intelligently extracted and abstracted information is stated to be useful when depicting relevant issues to users (Seppälä and Salmenperä 2005). When the process is running steadily the tools should support monitoring on a high and abstract level, e.g. enable energy efficiency to be analysed, but also enable lower-level information to be accessed for detailed examination. Then, on the contrary, when the process is in a highly changing state, the monitoring tools should enable the user to access more detailed aspects of the process. This may be seen as an information system-related requirement of flexibility, which is opposite to the automation system effectiveness requirement, as noted also by Paunonen (1997).

Maintenance is an important supporting activity of the whole production process. Its aim is to keep the system in a production state, and so to increase the total efficiency of the industrial production process. In general this is achieved with information processing, which humans and computers use together to control the reliability of machines in the process environment. The maintenance itself may be of either a proactive or reactive type. In proactive maintenance, typically, a model describing the reliability of the process and the related gear is built. In reactive maintenance models are not required. However, in both types of maintenance successful failure identification is required (Honkanen 2004).

In model-based approaches the processes have to be modelled to some extent and, as systems grow in size and general complexity, this model also gets more complex. If more autonomous operation is required, then a more descriptive model of the system is needed. These models may contain information about the physical connections of quantities, process-related physics, optimisation principles, and process control philosophies. In general, rather extensive and descriptive models are needed to get something intelligent to happen automatically, but this is seen as problematic because the modelling fast becomes laborious. For example, the operator support system for

the mineral filtration process presented by Jämsä-Jounela (2005) contains a complex mathematical model which has numerous *assumptions*, at least three *ifs*, multiple parameters that *can be* calculated, at least four *empirical* constant parameters, and a couple of parameters that are to be identified online. This model-based approach has proved to be useful in the particular case of the filtration process for optimisation and fault diagnostic purposes. Although the developed computational system is modular in structure, it could be argued that the presented concept is hardly highly usable in a totally different process setup or when the current process changes substantially.

2.5 Future trends for process automation development

Available technical achievements have been the driving force in the selection of what new possibilities have been offered to users and developers in process automation. With digital communication and almost fully software-based operation there are no big barriers any more to what functionalities may be developed and thus offered to users. This will result in a whole new set of functionalities and services for users working with process automation; e.g. Ventä (2005) has gathered the generally recognised new possibilities (Figure 2). However, from now on the selection of what will be developed will be guided by other limiting issues, such as economic preferences, quality and safety requirements, and, of course, the selection made by users.

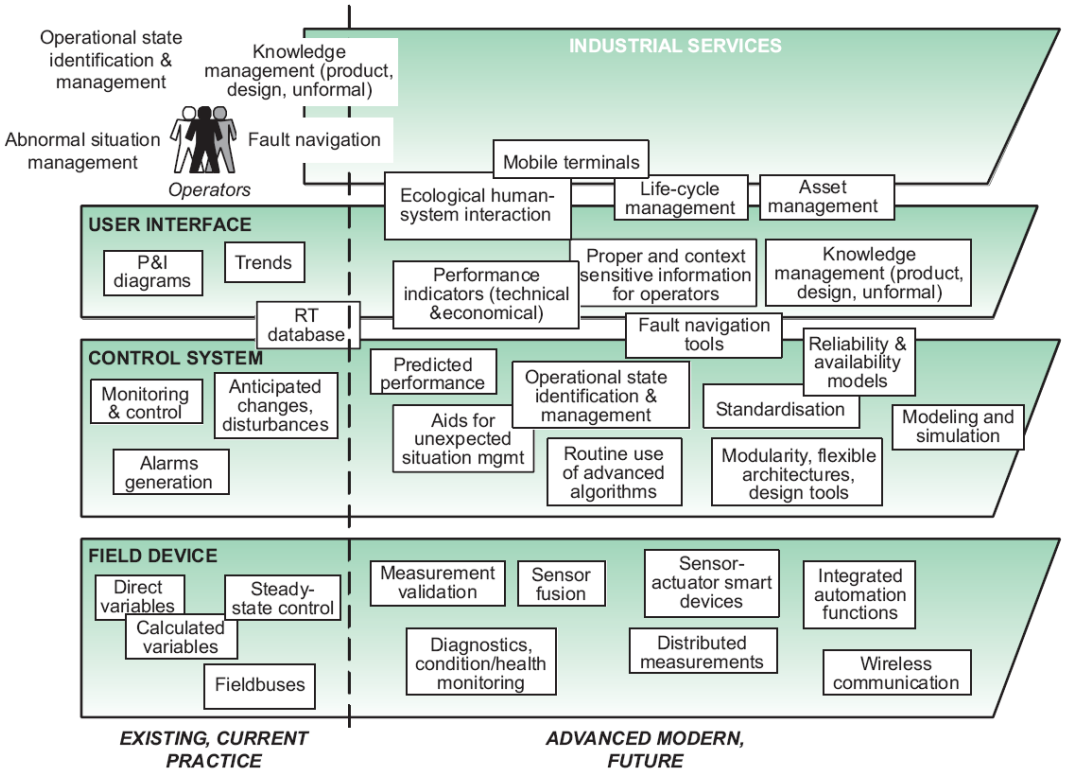

Figure 2 - Intelligent automation technology roadmap (adopted from Ventä (2005)).

The realisation of the possibilities visible in Figure 2 requires different know-how than previous developments in process automation, limited mainly by hardware capabilities. Mastering a variety of software methodologies and knowledge tools is becoming an important property for developers in process automation (Ventä 2005). Furthermore, wireless communication is becoming reliable enough for other than real-time control operations, and it may be used for supporting operations in information

processing. Especially for monitoring this provides interesting new possibilities, as cheap embedded electronic modules that communicate in a wireless manner could be added to field devices to gain information for diagnostics and monitoring functionalities. In addition, increasing the number of measurements could be used to compare different measurement sensor outputs (Elmusrati et al. 2007; Ventä 2005).

New opportunities do not just affect the normal operation of systems, but changes are also reflected in development phases. Although these phases in process automation are comparable to ordinary software and hardware development projects, the development and programming in automation was rather isolated in the past. The requirements of real-time responsiveness, reliability, and the need to be able to realise control philosophy have kept design methods isolated. This isolation was criticised by Olsson as early as in the year 1992, but at least the recent maturity of general electronic hardware and software engineering tools has made it necessary to check if this separation is still appropriate (Heck et al. 2003; Rockwell 2007; Sierla et al. 2007; Ventä 2005;Wagner 2002).

Business trends have also changed in process automation, resulting in growth in the responsibilities of individual humans working in industrial sites. As production systems have become more integrated, because of efficiency demands, there are more individual items to watch over and users are working more often with issues that they are unfamiliar with (Jämsä-Jounela 2007). Maintenance is more extensively outsourced, resulting in users diagnosing systems that are working in unknown environments more often (Pyötsiä 2005; Theiss et al. 2007). The rise in the amount of information available is also the result of technical development, as there are no limitations on what data may be stored and transferred from the measurement to the system level. But more unprocessed data actually make the operators' situation worse and, to battle this trend, the user should have tools that help them to find relevant information faster and get help when making decisions in more complex situations.

New business requirements also affect the way in which processes are to be controlled in the future, thus requiring new functions from the control systems. Table 1 lists the new requirements that have been reported lately (Chokshi 2005; Chokshi and McFarlane 2002; Chokshi and McFarlane 2008; Jämsä-Jounela 2007; Keller and Bryan 2000). As these requirements affect the way processes are controlled, these also affect the requirements for monitoring systems. The following table includes a column done by the author that specifies properties that seem to be required from monitoring systems as a result of changed control requirements.

Table 1 – New requirements from business trends setting up requirements for control and monitoring systems.

Business-driven requirement	Requirements and effects towards control systems	Requirements towards monitoring systems
Flexibility and volume variation	Being able to use process instrumentation in multiple ways	Rule adaptation that is able to adjust itself to changed process settings and values
Responsiveness and process safety	Faster change management (e.g. campaign mode) and response to disturbances	Methods and system structures able to respond to frequent changes and distinguish between controlled changes and disturbances. Better and extended coverage of algorithms is needed so as to have timely response to disturbances
Material cost reduction	Using less materials in addition to being able to use cheaper raw materials (that are not of such good quality)	Responsiveness in monitoring as increased variation in materials increases the risk of blockages in the piping which need to be observed in time.
Energy efficiency	The energy efficiency is typically raised, utilising complex interconnections in the process (e.g. using Heat-Exchange Network)	Handling the complexity; interconnections are problematic, especially in change situations.
Capital cost reduction	Use the instrumentation more efficiently (e.g. by increasing speed or reaction temperature) and reduce inventory sizes. For example, utilisation of Vendor-Managed Inventory requires accurate estimates of future consumption of materials.	Using instrumentation functioning at its operational limits is more likely to cause all kinds of disturbances. Reduced inventory requires better tools to calculate the timing of changes in the production chain.
Increased product quality	Better and typically more interconnected control algorithms lead to less variation in production.	More complex control is more sensible to disturbances and sudden changes. Tracing problems in the process stream is needed.

Similar requirements have been motivating holonic research related to manufacturing control (Bussmann and McFarlane 1999; McFarlane and Bussman 2003). However, recent studies reported by Schild and Bussmann (2007) have argued that an enormous amount of flexibility is not really required in real industrial settings. In addition to requirements set by business considerations, the general monitoring system requirements should be taken into account as far as they affect system-level development; for example, Venkatasubramanian et al. (2003c) list these as including responsiveness, adaptability, and parallelism.

Limitations and problems of the current development trends and approaches:

With respect to supporting process automation functionalities, systems have been suffering from decisions that were made in the past, e.g. alarm handling and processing is based on a rather simple approach developed when the first digital control systems were introduced. Devices and control systems, as well as their monitoring, have been built for continuous and steady usage, and, further, their operation has been optimised for a steady state. In terms of monitoring, enough attention has not been focused on operations in change situations, ensuring their correct operation and further optimisation. These situations have been typically treated as special cases handled with special attention (e.g. by using enough personnel). However, this is not the case any more, as systems have become larger and more complex, and there are not enough personnel available on-site to handle more common change situations with special attention.

The currently probable and possible future trend related to monitoring goes towards specialised individual software systems that are each built for a certain task. Variation exists between the branches of process industry about how much diversity in solutions there will be. Nevertheless, the base techniques used have become more coherent between solutions, thus making integration in theory an easier task. Furthermore, typically each individual software solution supplier built their system by just configuring an already available commercial base system. A characteristic example is industrial reporting systems that currently are typically built with a web-based user interface accessing data stored in a standard SQL database. Customisation for customers' needs is done by the implementation of case-specific terminology and the selection of appropriate application modules. Typically, the integration of numerous solutions is realised as an application portal that enables different solutions to be accessed relatively easily. The result with this is that the user is restricted to the links provided by the system engineer, and flexible customisation based on dynamically changing process values is not supported.

On the technical side today's systems rely heavily on centralised and specialised database approaches, e.g. every solution has its own server running proprietary data schema. The operational principle is user-driven, via a web browser or Windows client. One part of the software is responsible for fetching data from an external system, (e.g. with a field bus from devices, OPCs, or people inputting values with reporting forms) and then another part is used to fetch information relevant to the current needs of the user. If the active functions are monitoring value changes, these are typically organised as separate client applications, because databases are not, in principle, designed for running periodic value checks. However, stored procedures in databases may be used to realise this, but they have performance limitations and may not be responsible enough to monitor applications in process automation. Furthermore, algorithms in monitoring are typically structurally static and their operational logic is developed in the system design phase. In addition, control algorithms and even monitoring algorithms usually fail if some part of the data is missing. This may happen easily if some device is temporarily out of order, unavailable (e.g. with wireless devices), or the system setup has been updated.

The conclusions of current trends and approaches may be characterised as follows: the technical limitations of the past have resulted in static structures and operating principles. Applications have become multi-functional, but the functionalities are specified at the time of design by the engineers. Users are able to modify parameters but not the operational principles, which hardly results in sufficiently flexible operation. The structurally static operation is also visible in the algorithms where, typically, only the numerical values change and not the way things are checked. The strong emphasis on reliability requirements has resulted in systems that, especially in monitoring operations, are not sufficient any more as the size and complexity of processes have grown substantially.

2.6 Conclusions about monitoring

In this thesis the objective is to research and develop flexible system solutions for human users to perform monitoring tasks in the process automation environment. On the basis of the discussion in this chapter about development trends in technical and business perspectives, a new monitoring system should have the following general properties:

- **P1: Flexibility** – system should be usable for a variety of tasks, and adaptation to process changes needs to be supported (Jämsä-Jounela 2007; Keller and Bryan 2000). Furthermore, flexible personalisation of capabilities is required in order to support human decision-making (Venkatasubramanian et al. 2003b).
- **P2: Delegation** – because of the size and complexity of current and future production environments, the users' work tasks should be supported with automation as much as possible, e.g. with delegation (Maes 1994).
- **P3: System integration** – no one system or method is alone sufficient to provide all the information that is needed for success in monitoring tasks. In addition, relevant information for monitoring will be available in a multitude of distributed legacy data sources. Thus, integrating multiple systems are required to overcome the limitations of individual systems, e.g. by utilising hybrid systems (Venkatasubramanian et al. 2003b).
- **P4: Knowledge handling** – information is stored and available in various data formats and the monitoring system should support the unification of these. Further, users should be able to describe the meaning of things and their relations.
- **P5: Data processing** – the system should be capable of performing data processing to extract and abstract relevant information from vast amounts of data, which in processes are available in a time series format. In addition, systems should assist the user to find useful relations in the data (Gentil 2006).

The technological tools that may be used to develop systems to support these properties are discussed in the next chapter.

3 Agents, semantic web technologies, and their application to monitoring task in process automation

3.1 Introduction

In process industries the traditional barriers to adapting solutions from Information Technology (IT) to automation technology are gradually disappearing. In this changed situation a new type of know-how will be required to succeed in the development work and mastering software-related methodologies and tools will be important, in addition to traditional hardware- and electronic-oriented skills. In monitoring tasks the state-of-the-art technologies will have great potential to help bypass the limitations of the current approaches, e.g. the structurally static operation discussed at the end of Chapter 2 is not able to adopt the full potential of distributed processing and wireless communication. Monitoring functionalities are not restricted by the reliability limitations of control operations directly modifying physical quantities, and thus offer suitable ground for realising the benefits of new technology. The objective of this chapter is to study the properties of agent technology so as to be able to select feasible structures and operational principles in the following chapters. In addition, some selected state-of-the-art technologies that currently have the strongest industrial drive within the integration and information-handling area are introduced.

Agent technology has been proposed as being suitable for dynamically changing distributed environments by researchers in computer science (Ferber 1999; Jennings 2000; Russel and Norvig 2003; Weiss 1999). The application of agent technology to process automation has been motivated by the wish to match the properties of the domain and the technology (Chokshi 2005; Parunak 1997; Parunak 1999; Seilonen 2006). Its application to real industrial settings has been at least demonstrated in numerous functions, and it has been suggested that it is especially suitable in resource allocation functions with planning and simulation (Pechoucek and Marik 2008). Although monitoring in process automation seems to be a potential agent application area (Gentil 2006; Wagner 2002), relatively few industrial studies have been reported (Bunch et al. 2004; Buse and Wu 2007; Gentil 2006).

From the software engineering viewpoint, agent technology may be seen as a promising new and advanced version of object orientation (Luck et al. 2005). Developments in software engineering methods have made it possible to handle more complex structures and the level of abstraction in programming languages has been rising (Soukup and Soukup 2007), and agent technology is one result of this development. Within the industrial sector the Service-Oriented Architecture (SOA) has gained a lot of interest and it offers properties (loosely coupled, modularised, and service-based operation) comparable to agent technology. In addition to comparing tools for organising system-level development, knowledge representation and data-processing issues are discussed. Strong interest in semantic technologies has appeared lately, because the internet suffers from incompatibility problems arising from the miscellaneous knowledge representations used by different services. Semantic research is trying to overcome this by developing machine-processable information formats.

3.2 Agent technology

Agent technology, or orientation as you could also call it, is a branch of computer science in which a system is thought of as containing pieces of software that act on behalf of the user or other software programs. Although having similarities with object orientation, the key difference of agents is that they act autonomously on the basis of their internal operating logic and have some level of control over their operation (Jennings 2000; Parunak 1997; Russell and Norvig 2003; Weiss 1999). Furthermore, agents are in general thought to operate in some environment, perceiving it with sensors and making modifications through actuators. This setup may be seen in Figure 3, and it should be noted that it bears many similarities to the system setup in which process automation monitoring systems operate.

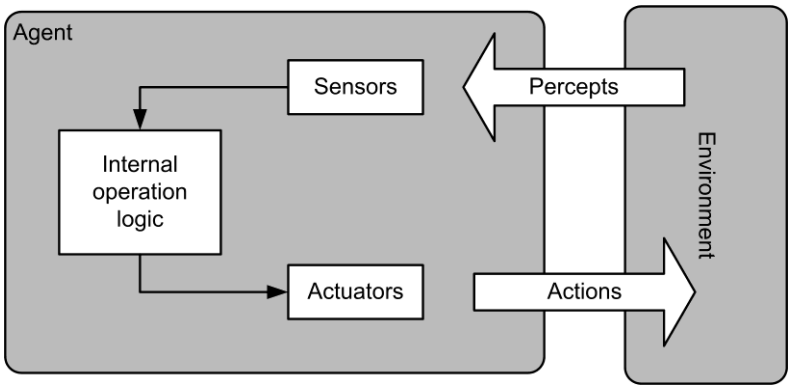

Figure 3 – Agent interacting with the environment. Modified from Russell and Norvig (2003).

To be able to benefit from using agent technology as a software engineering paradigm, it is important to understand its characteristics and properties. Although the exact definition of the concept of agents is work in progress, most of the researchers agree that agents are *rational software entities* having properties such as *autonomy* and *flexibility*, and that their operation is typically *goal-oriented*. The list of properties varies, depending of what kinds of agents are being discussed (intelligent agents, autonomous agents, distributed agents, mobile agents, etc.) (Ferber 1999; Russell and Norvig 2003; Weiss 1999). In addition, a typical agent-based system has a number of these active entities and is thus called a multi-agent system in which these entities *communicate* with each other.

Typical properties of agents are:

- Autonomous – agents have control over their operation; they can decide when and how to take actions.
- Situated – agents operate in some environment; they are possibly capable of perception and taking actions that modify the environment. The environment may be physical or computational.
- Reactive – agents respond to changes in the environment
- Proactive – as intentional entities agents act to achieve goals
- Flexible – agents have multiple possible ways of achieving goals
- Robust – agents recover from failure
- Social – agents interact with other agents

Having these properties, a system constructed with agents is said to significantly enhance our ability to model, design, and build complex, distributed software systems (Jennings 2000). This is stated to be mostly the result of the higher level of abstraction that the software engineer is utilising when working with agents (Padgham and Winikoff 2004; Viroli et al. 2007), and also with the help of the agent approach supporting flexible decomposition and organisation techniques (Jennings 2000). Because agents are by definition autonomous, they may be set to deal with smaller parts of a whole problem separately, yielding natural support for decomposition. However, agent technology has its roots and also limitations in Artificial Intelligence (AI), and it is mainly designed to operate with symbolic data (Kaplan 1984). This symbolic processing background should be taken into account, especially when adapting an agent technology to the process automation environment, where a major part of the data is numerical. Because of this agents should be used only for those functionalities to which they are best suited (Wooldridge and Jennings 1999).

3.2.1 Multi-agent systems

When adopting agent technology for real-world problems, it soon becomes apparent that an approach with multiple agents is needed (Ferber 1999; Jennings 2000; Weiss 1999). Much of the promised flexibility of agent technology is based on the dynamic organisation of agents and versatile communication that is operable despite changing structures. Figure 4 illustrates with an example how agents in a multi-agent system (MAS) may be set to be responsible for areas of the environment, and how an agent society has multiple layers in its structure. The agents in the society exchange information either directly via messages or indirectly via changes in the environment. As messages are more controllable, they are typically used.

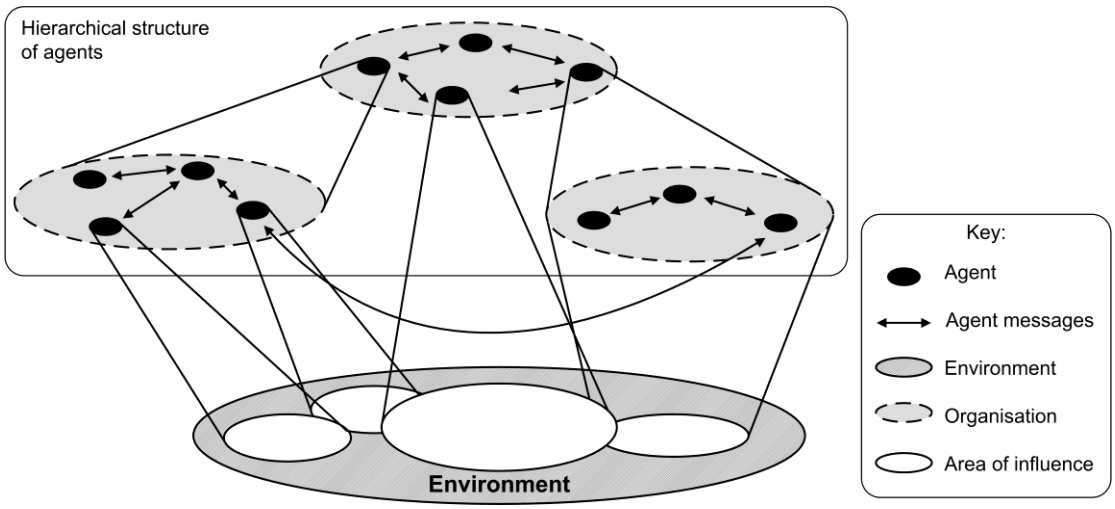

Figure 4 - Agent organisation on multiple levels, each having an area of responsibility.

Sophisticated negotiations are an important benefit of using agents to construct software systems. Negotiations are constructed from series of messages that agents send to each other. Typically, agents try to coordinate their actions with negotiations, but a competitive approach is also possible. The Foundation for Intelligent Physical Agents (FIPA 2008) has standardised agent negotiations based on *speech act* theory, originally introduced by John Earle in the year 1969 to imitate communication between humans. FIPA defines various types of messages or communicative acts, each having a specified role in communication. Furthermore, these communicative acts are combined together to form a whole negotiation, called an interaction protocol.

Figure 5 illustrates a standardised *fipa-subscribe* interaction protocol, which is used to delegate change detection (FIPA 2002). The standardised communicative acts and interaction protocols are general in agent technology and their usage is not restricted to any specific application area.

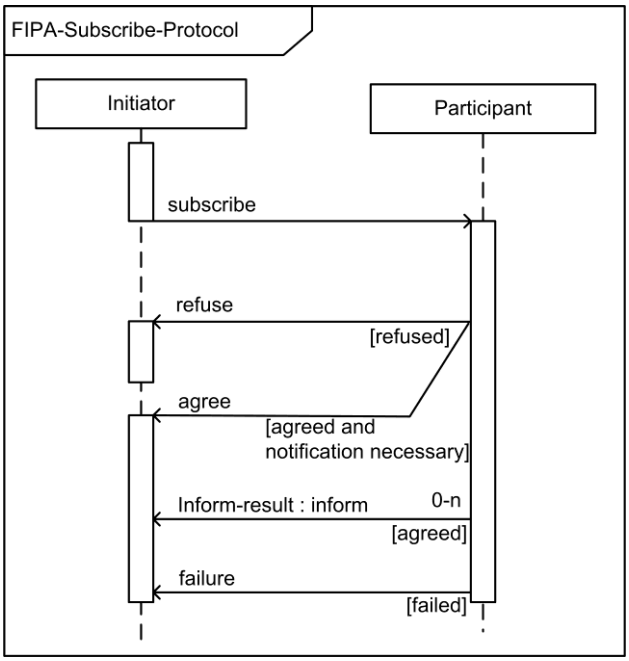

Figure 5 – Standardised fipa-subscribe interaction protocol (adopted from FIPA (2002)).

Although negotiations have been standardised, the selection of an appropriate level of interaction and selecting the correct type of structure for the agent society is left to the user. A lack of structure and control may result in a system with unwanted emergent behaviour (Wooldridge and Jennings 1999). Currently, a lack of effective tools for debugging agent society interactions has been noted (Poutakidis et al. 2002). Bordini et al. (2007) have suggested that instead of defining agent negotiations on a message level, agents in a system should be defined with roles defining responsibilities and the interactions should be executed on an intentional level to support the validation of correct operation.

3.2.2 Agent operating principles

In their operation, agents may be divided into reactive, deliberative, and hybrid types. The operation of a reactive agent is based purely on current percepts. Their simple operating principle makes this type especially responsive and suitable for real-time functions. However, constructing complex operations with purely reflexive operation without memory easily becomes problematic because the number of rules easily becomes unmanageable (Russel and Norvig 2003; Wooldridge 1999). Alarms in process automation may be seen as being closely related to this type. Reactive agents are reliable but active adaptation to changing situations is hard to build into them.

Deliberative agents are capable of pursuing long-term goals with a set of actions (Dickinson 2006; Jennings 1999; Padgham and Winikoff 2004). Deliberative operation has been stated to be suitable for complex operations in dynamically changing environments (Jennings 2000; Dickinson 2006; Russell and Norvig 2003; Weiss 1999). A currently popular (Bordini et al. 2007; Dickinson 2006; Helin 2003;

Jennings 1999; Jennings et al. 1998; Padgham and Winikoff 2004) way of realising this is making agent operation intentional with the Belief-Desire-Intention (BDI) agent model (Rao and Georgeff 1995). Figure 6 illustrates layout and data flow in a generic BDI agent architecture.

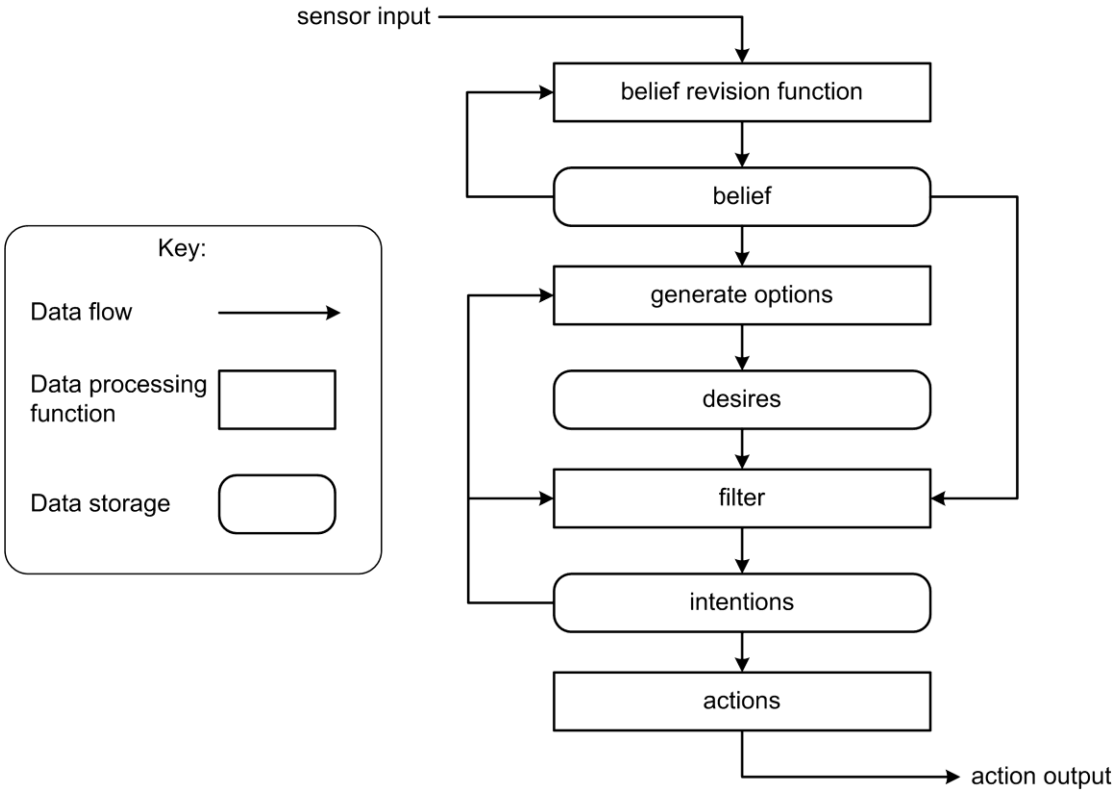

Figure 6 - Generic Belief-Desire-Intention (BDI) architecture. Modified from Wooldridge (1999).

Generic BDI operation has the following phases (also visible in Figure 6):

- *belief revisions function* - take the input from the physical or software environment and process information for further usage. The processing is typically influenced by the current understanding of the state of the world (beliefs).
- *belief* - database storing the agents' current understanding of the world.
- *generate options* – on the basis of the current state of the world and operational situation, select the desires that it is possible to achieve with the available tools. This provides context sensitivity in the operation.
- *desires* - expresses the end results that the agent is currently pursuing. These are the tasks and respective objectives that the agent should achieve in the long run.
- *filter* - select from multiple end result candidates the ones that are currently most worthwhile to pursue.
- *intentions* - a set of more or less concrete sub-objectives that the agent has decided to try to achieve with a set of concrete actions.
- *actions* - these are the actual doings that somehow modify the environment and are going to shift the agent closer to the desired end state.

In actual software realisations this BDI model is typically transformed into beliefs, goals and plans (for example, in Bordini et al. 2006; Huber 2000; Jack 2008; Jadex 2008; Jason 2008). In these realisations, the plans are concrete procedural descriptions of sets of actions that are to be performed in order to reach a certain goal (desire in the general model). Agents utilising a BDI model to process diagnosis functions have been proposed by Ingrand et al. already in year 1992. Furthermore, the BDI model has been stated to be especially suitable for constructing agents that process information available on the Semantic Web on behalf of a human user (Dickinson 2006). Lately, BDI model-based agents have been suggested for information processing and condition monitoring in power systems (Buse and Wu 2007).

In real-life applications the actual agent architecture is most often a mixed combination of reactive and deliberative types. Typically, this is realised with either a vertically or horizontally organised layered architecture in which decision-making is performed in several layers, each operating on a different level of abstraction. In horizontal layering each layer is directly connected to input and output and operation is performed in parallel. Figure 7 illustrates a two-pass version of a vertical architecture, where the lowest-level layers are directly connected to the environment and information to the upper layers is flowing through the lower layers (Wooldridge 1999).

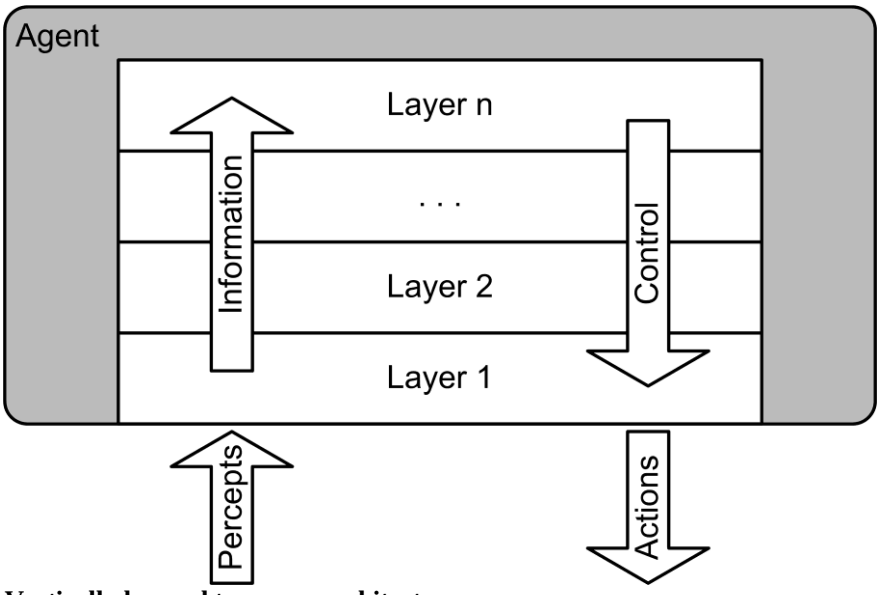

Figure 7 - Vertically layered two-pass architecture.

The vertically layered two-pass architecture operation has similarities to organisations; the lower levels provide abstracted information to the upper layers and, after decisions are made in the upper layers, the control flows down through the layers (Wooldridge 1999).

3.2.3 Information processing with agents

It has been stated that if data and resources are distributed, or if a number of legacy systems must be made to work together, then agents are an appropriate technology for building such a system (Wooldridge and Jennings 1999). On the one hand, proactive agent operation provides the possibility of delegating the information retrieval and processing tasks (Dickinson and Wooldridge 2005; Maes 1994; Tennenhouse 2000),

and agents could be used as secretaries or butlers performing information-related tasks (Negroponte 1995). On the other hand, the rationality of agents and flexible organisation of multiple agents are beneficial when accessing and retrieving information in dynamic and distributed environments (Decker et al. 1995; Huhns et al. 2005). Because agents seem to have potential in information-related operations, the concept of an information agent has been introduced. Klusch (2001), for example, defines an information agent as an agent that has access to one or multiple, heterogeneous and geographically distributed information source(s), and it proactively acquires, mediates, and maintains relevant information for the user.

Typically, information agents use match-making and brokering techniques in information task performance (Klusch 2001; Klusch et al. 2003; Sycara et al. 2003) and the utilisation of semantic information in these processes has been demonstrated (Nodine et al. 2003). In addition, a concept close to that of information agents is that of a wrapper agent that *wraps* some data store that has a possibly unknown internal representation format, and thus this wrapping enables e.g. information agents to access data stored in a legacy data store.

Information agents have been demonstrated in industrial applications. The ARCHON system (Cockburn and Jennings 1995) wrapped pre-existing industrial systems with information agents to gain fused diagnosis information, and similar ideas have also been proposed for power systems by other researchers (Bann et al. 1997; Buse el al. 2003; Mangina et al. 2001). Related to monitoring in process automation, information agents have been proposed as being suitable for communicating alarms to operators (Bunch et al. 2004), the flexible definition of alarm conditions (Koskinen et al. 2003), and the analysis of measurement data for diagnosis (Gentil 2006; McArthur et al. 2005).

3.2.4 Agent-oriented software engineering

To be able to exploit the benefits of agent orientation, the engineering methodologies used must be suitable for agent development, and Agent-Oriented Software Engineering (AOSE) methodologies are designed for this, similarly to the way that the Unified Modelling Language (UML) and Rational Unified Process (RUP) method are for Object Orientation. The selection of AOSE should be related to the selection of the operational principle of the agent. For rational and goal-based agents the currently relatively mature AOSE methodologies are Gaia (Zambonelli et al. 2003), MaSE (Wood and DeLoach 2000), Tropos (Castro et al. 2001), and Prometheus (Padgham and Winikoff 2004), according to a listing provided by Bordini et al. (2007). From a broad perspective these methodologies are similar, as they all define at least a loose step-by-step procedure that may be used by the software designer and all cover agent role definition and interaction design in a similar way, but the levels of maturity and practical examples differ.

Prometheus methodology:
By the time the research behind this thesis had been carried out the Prometheus methodology seemed to be the most mature and to offer the best support for practical usage. While most of the other methodologies had more or less disunited documentation, a quality book covering the Prometheus development process in enough detail and with practical examples was available (Padgham and Winikoff 2004). Prometheus is designed for BDI-type agents and supports the architectural

design phase of the whole process especially well. Prometheus is rather new, and thus utilises the latest knowledge in software design issues, and was developed together with one of the most advanced companies utilising agent-related software in commercial applications (http://www.agent-software.com.au/). In addition, the use of the Prometheus methodology has had a significant impact on how well students have been able to utilise agent systems in their projects (Bergenti et al. 2004) and there is also tool support currently available for the methodology (http://www.cs.rmit.edu.au/agents/pdt/).

The Prometheus methodology provides a systematic step-by-step procedure and specifies the issues that are to be covered in each design step. Figure 8 gives a general overview of the methodology and illustrates the development phases and design artefacts of Prometheus. The Prometheus methodology categorises the development into three major design phases, aligned vertically in Figure 8.

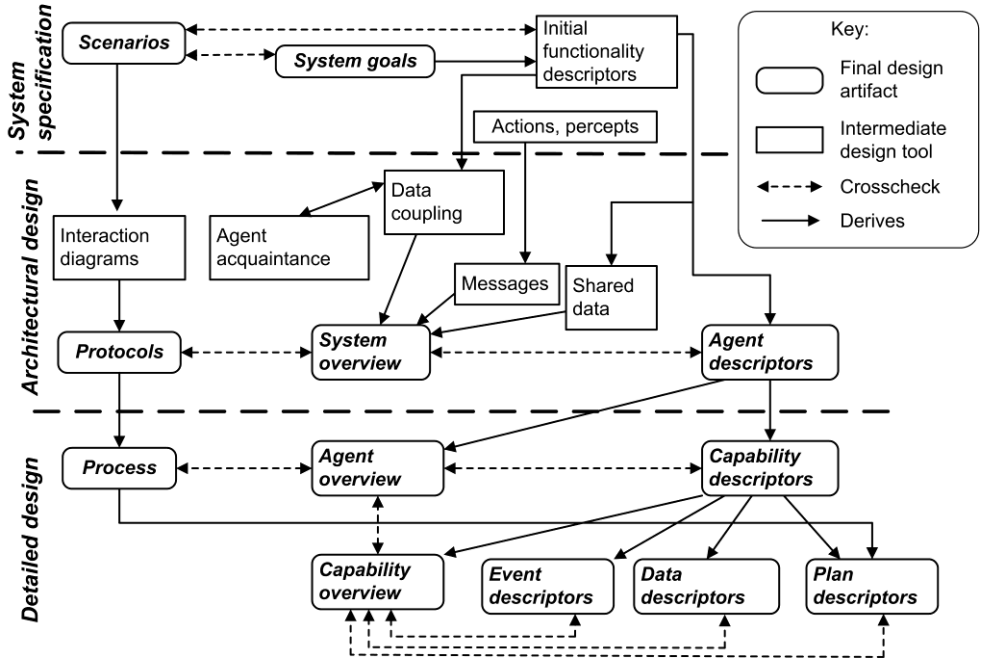

Figure 8 - The phases of the Prometheus. Adopted from (Padgham and Winikoff 2004).

The practical guide for Prometheus from Padgham and Winikoff (2004) lists the design phases and their major deliverables as the following:

System specification is the initial phase of the development, and it focuses on identifying the basic functionality and requirements of the system with the following deliverables:

- System goals – identifying goals describes the operation of the system on a high level, listing what individual results the system provides to the user.
- Scenarios – The system operation is concretised by defining process-oriented descriptions of it, similar to UML use cases.
- Functionalities – identifying the basic functionalities of the system with short descriptions. Functionalities group one or more goals together and specify related activities that are needed to form certain behaviour.
- Action and percepts – An overview of agents' percepts (inputs) and actions (outputs) in the form of a list describing how the system interacts with the environment and what artefacts are to be used.

The ***architectural design*** phase is used to define the general agent system architecture, which is used to meet the defined requirements. This phase has the following deliverables:

- Agent types – listing the agent types in short form. A descriptive name for each agent type is a good starting point on the overview level.
- Protocols – define a suitable interaction protocol for each agent functionality. The dynamic behaviour of the system is heavily based on these protocols. For protocol diagrams Prometheus uses AUML (2008).
- System overview – defines the general static system structure.
- Agent descriptions – defines agents' responsibilities, similar to UML class definitions.

The ***detailed design*** phase focuses on individual agents and their capabilities, e.g. plans describe how the agent is supposed to achieve its goals with processing events and data. This phase is specified to result in the following deliverables:

- Agent overview diagram – describing in further details what functionalities each agent has and how these may be combined into clusters (capabilities).
- Process specifications – defines the interfaces of individual agents and how the agent processes these external requests to create functionalities.
- Capability descriptors – clusters of functionalities and their descriptions.
- Capability overview diagrams – figures of capabilities and their relations.
- Plan, data, and event descriptors – the final phase of the design, resulting in detailed descriptions for the implementation of each design artefact.

The Prometheus methodology specifies the whole development process, but it has been stated that it may be adapted for specific uses by exploiting only selected parts of it and using the methodology more like a set of guidelines (Padgham and Winikoff 2004). The experiments that were conducted (Chapter 6) provide a further demonstration of the use of the Prometheus methodology and its design phases and the use of artefacts.

Although Prometheus supports the use of agent concepts in design, it does not facilitate integrated and automatic development or the construction and validation of the actual system. Bordini et al. (2007) have stated this major drawback to be common to all of the currently available agent methodologies. Furthermore, it is noted that currently the design is manually implemented, and further that there is a need for practical tools that support the verification and validation of the realised agent system. It is likely that actual agent systems will be implemented, at least partly, with object-oriented techniques (Wooldridge and Jennings 1999), and that agents and objects as methodologies will be seen to complement each other (Odell 2002). It has also been stated that augmenting traditional techniques with agent technology could be a useful approach in the adoption of the benefits of agents (Luck et al. 2005).

3.3 Service-oriented architecture

Service-Oriented Architecture (SOA) is a paradigm that is agreed world-wide, promoting interoperability when organising and utilising distributed business capabilities under different ownership. In SOA the idea is that business offerings are described as services and then participants use discovery methods to find these

services. Furthermore, guidelines as to how the service provider and consumer interact are provided (Erl 2005). An ideal SOA provides modularised and loosely coupled interoperability between services running on mixed operating systems and implemented with various programming languages distributed over business networks. Currently, the most prominent concrete implementation of SOA seems to be the XML-based SOAP (W3C SOAP 2007) realising Web Services (W3C WS 2002), but others too, like the more traditional RPC, DCOM, and CORBA techniques may be used. However, actual specifications and operational principles are needed to make services interoperable in practice. The W3C-originated Web Services (W3C WS 2002) are probably the most commonly used implementation of SOA, and the competing reference model OASIS SOA (2006) is also available.

Although the service profiling and interoperability are rather mature in SOA implementations, there is no clear concept of how to automate the composition of the service. The current practice for this is that the use of services is predefined by humans. Orchestration describes how the coordination of services takes place from a single consumer's viewpoint and the current industry standard is BPEL (OASIS WS-BPEL 2007). Choreography, on the other hand, defines rules for how different peer-to-peer parties operate together, e.g. to form more complex services. Choreography is a more complicated research issue, and there the standardisation work is currently in progress (W3C WS-CDL 2005). Furthermore, it has been stated that the development of service-oriented systems could utilise advances available in multi-agent research, especially those related to autonomy and flexible collaboration (Huhns et al. 2005). In addition, it has been stated that agents are actually suitable and thus should be used for the higher-level control of services (Dickinson and Wooldridge 2005; Lassila 2007).

3.3.1 Web services

Web Services are currently a commonly used strategy to implement the ideas that SOA provides. Web Services is a set of specifications originating in W3C that concretely describe issues related to service provision and access (W3C WS-ARCH 2004). There the service provider first describes services with Web Services Description Language (W3C WSDL 2007) and requests Universal Description, Discovery and Integration (OASIS UDDI 2004) to store and publish this information. Then the services may interact in a stateless manner with each other using SOAP (W3C SOAP 2007). These three specifications enable services to interact on a basic level, but it does not describe how individual services are carried out to form integrated complete business process cases. Therefore, Web Services Business Process Execution Language (OASIS WS-BPEL 2007) has been developed.

Although Web Services are becoming the industry standard, it is argued that they fall short in providing improved automation and interoperability, as the user always needs to configure the operation (Lassila 2007). Furthermore, as Web Services are gaining strong industrial acceptance, there has been interest in how to integrate agents and Web Services. Actually, this integration has been seen to be possible in both ways; agents have been utilising Web Services and also Web Services have been using the functions that agents provide (Greenwood et al. 2007).

3.4 Semantic technologies

Semantic technologies provide tools and mechanisms that may be used to describe the data in such a way that software programs can utilise it in multiple ways. This is useful in software engineering, e.g. separating decision-making rules and declarative data definitions facilitates the easier development and maintaining of systems (Lassila 2007). The opportunities provided by semantic technologies are great when integrating a multitude of systems together (Ciocoiu and Nau 2000), which is the aim of the developers of the Semantic Web (W3C SW 2008), currently the most prominent realisation of semantic technologies.

In addition to the above, the possibilities of semantic technologies go beyond defining the meanings of the actual data. As Umberto Eco wrote in his famous book Foucault's Pendulum; "*No piece of information is superior to any other. Power lies in having them all on file and then finding the connections. There are always connections; you have only to want to find them."* (Eco 1989; 225). Defining the relations and connections in semantic data models supports the use of these models for combining and reasoning purposes. If this modelling is done on many different levels of abstraction it may also be used to access, process, and find information, even in a case where only partial information is available. In addition, as semantic technologies aim to separate the actual data processing and the use of descriptive models, it is a concept that supports end-user and do-it-yourself life-cycle development methodologies.

Although the actual technical realisation of *semantic technologies* is nowadays often based on the Semantic Web, the issue itself is much wider. The actual languages and tools used for semantics need to be defined before semantic technologies can be used with a software-based system. This definition includes the languages used and providing concrete syntax, but the terminology describing things in the domain is also needed. Although this chapter introduces Semantic Web-related techniques, the benefits of using declarative data definitions are also usable with other tools.

3.4.1 Methods and tools for ontology engineering

Looking up a definition for the word *ontology* in the dictionary does not help the ordinary software developer but the definition used in computer science is "specification of conceptualisation", which is actually a little more understandable. And the term ontology is even easier to understand if one thinks of it as data modelling, albeit executed in a new format and with a new set of software tools. However, data modelling may be seen as having a tighter connection to the purpose of the model and to the tools and techniques used, as an ontology could be stated to be more objective by definition. And if data modelling is hard, ontology engineering is even more problematic (Gavrilova and Laird 2005).

Ontology engineering may be performed in a variety of ways, and ad hoc might be the most typical. In this "approach" the respective group of developers and users just document the domain knowledge with suitable ontology tools. One such editor for realising ontology engineering is Protégé, which supports numerous actual ontology languages (Protégé 2008). To be more controlled, the actual ontology engineering may be organised. For example, when developing ontology for some specified purpose controlled discussions with the interest group initiated by the moderator realising the specified procedure might be used (Tempich et al. 2007). Another possible way to develop ontologies is to first develop a base ontology with a huge

amount of terms. Then the ontology engineering is supported with a suitable infrastructure that facilitates the use of that base terminology (Hyvönen et al. 2008). These are just examples of how ontology engineering might be arranged, and a good level of domain knowledge is needed, regardless, to be able to produce usable and good-quality ontologies.

3.4.2 Semantic web architecture

The Semantic Web is about concretising semantic technologies in a World Wide Web environment to support data sharing and reuse (Berners-Lee et al. 2001). To enable this to happen, two things are needed: common formats for information interchange and languages for ontologies that describe data in a way that is understandable to both humans and machines. W3C (W3C 2008), the collaborative organisation promoting the Semantic Web, is developing a set of specifications that in combination may be used to respond to these demands. The developing process in the continuous and current set of standards and specification are visible in the W3C "layer cake", illustrated in Figure 9.

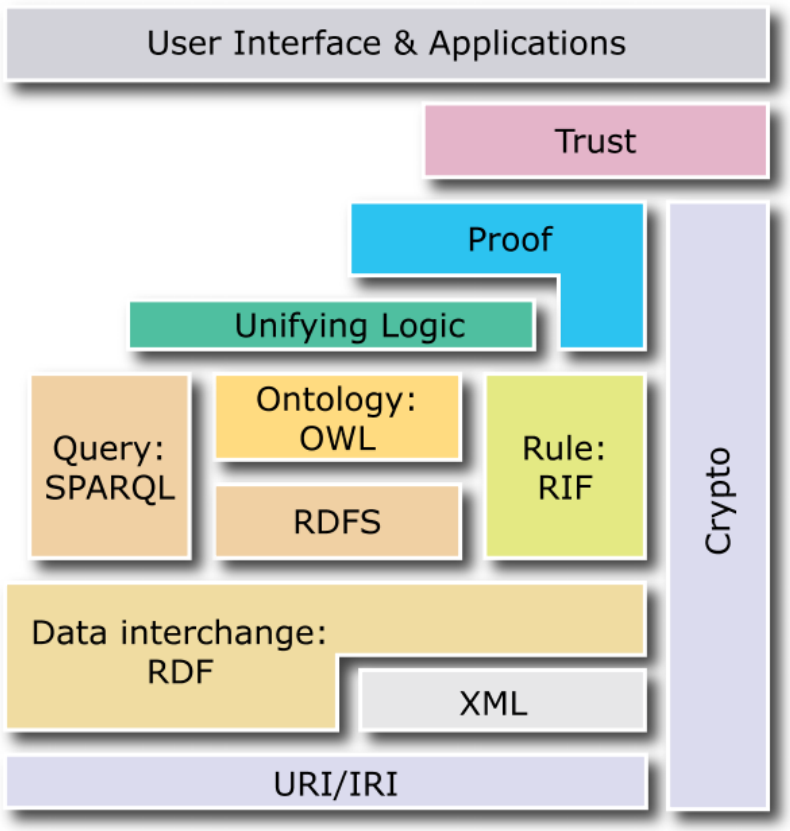

Figure 9 - Semantic Web "layer cake" (adopted from W3C SW (2008)).

The Semantic Web "layer cake" describes two things simultaneously. On the one hand, it shows how the whole system is made up of individual specifications, each responsible for some particular part of the overall functionality. On the other hand, the stack illustrates the current status of the standardisation work. If there is a name specified in a certain box then this part of the stack is standardised and is thus more mature than the boxes that do not have names specified yet. On the lowest level are URI (Uniform Resource Identifier) and XML (Extensible Markup Language), specifications for defining the syntax of the exchanged data. Then there are concrete

specifications for meta-data and semantics definition, for example OWL (Web Ontology Language), and further languages for rules and queries. Then further up in the stack there are more abstract issues of trust and proof which are built on top of concrete languages, and at the very top there are the user interface and applications.

In relation to knowledge representation issues, the most advanced standards provided by the Semantic Web are currently OWL (2008) and SPARQL (2008). OWL is a language for defining machine-processable ontologies storing declarative data, explicitly describing the meanings of terms and the relationships between terms. As terms in XML may have a tree structure, OWL may be used to define relations freely (graphs) and the vocabulary for describing relations has more variety. Basically, OWL enables classes and their relations to be defined in such a way that machine-performed reasoning is possible. SPARQL looks similar to the SQL language used to query data from databases, and it can be used to query graphs from an ontology defined in the OWL language (although originally developed for RDF). SPARQL can be used to query graphs matching a variety of pattern definitions and value constraints. Furthermore, it supports naturally diverse data sources and the use of mixed formats, as it can be used to make data uniform.

3.4.3 Semantic web services

Current specifications of the Semantic Web aim towards a situation in which individual services can exchange semantically understandable information with each other. However, it does not solve the problems of how to automatically discover, perform, and compose these individual services, leaving much of the service utilisation up to the responsibility of the user (McIlraith et al. 2001). Adding semantics to service descriptions is said to make it easier to take advantage of a service that only partially matches a request (Lassila 2007). Therefore, Semantic Web Services (SWS) aim to bring two different research areas together; namely Web Services and the Semantic Web (Martin and Domingue 2007a; Martin and Domingue 2007b). Both of these have already been discussed in this thesis (see Chapters 3.3.1 and 3.4.2).

The purpose of SWS research is to develop standards and methodologies to semantically describe services and enable the service composition to be automated further. The development of SWS is largely work in progress, and numerous standards are under development. The reader should refer to the work of Martin and Domingue (2007a; 2007b) and also the related W3C recommendations, e.g. W3C WSDL (2007) and W3C OWL-S (2004) standardisation work, for further information. However, it is interesting that the SWS-related W3C interest group was closed at the end of February 2008 (W3C SWSIG 2008). Furthermore, agent technology has been proposed as being suitable for performing service composition (Dickinson and Wooldridge 2005; Gibbins et al. 2003; Greenwood et al. 2007; Huhns et al. 2005), especially when building applications that are designed for human users (Dickinson 2006).

As a concrete W3C proposal the OWL-S service ontology definition covers the following three parts. The service profile (1) is mainly used to describe to humans what the service does. The process model part (2) describes how a client can interact with the service, which is crucial when enabling a machine to automatically deliberate on service selection and execution. This description includes the set of Input, Output,

Pre-condition and Effect (IOPE) of the service defined in machine processable way. The service grounding part (3) specifies practical technical issues related to communication, protocols and message formats. (W3C OWL-S 2004) However, the OWL-S defined IOPE model seems to have much similarities with agent based BDI-model, and forthcoming research will show how these two approaches are linked together.

3.5 Trend analysis in process automation

The previously discussed advanced information technologies are mainly designed to process information in symbolic form. However, process automation is an application area in which much of the interesting information describing continuous processes is in time series format. A time series is a set of records containing values with time stamps, and the order of these records is also of interest, as is the value itself. For example, the trend of temperature measurement is a typical type of time series data used in process automation. For the human user time series data may be visualised directly with trend graphs (Paunonen 1997). However, before the advances in agent and semantic technology can be fully utilised, significant features of the data in time series format should be extracted to symbolic form. Interesting features presented compactly in symbolic form would also be beneficial to humans monitoring processes as their responsibilities grow and supervising all the data in time series format becomes unrealistic (Seppälä and Salmenperä 2005).

There are numerous ways to create symbolic data from sensed inputs, e.g. using data mining, classification, and rule discovery techniques (Daw et al. 2003). An interesting phenomenon in time series data is change points, where the characteristics of the values change significantly, and these may be found with mathematical methods (Chopin 2007; Kundzewicz et al. 2000; Last et al. 2001; Takeuchi and Yamanishi 2006) Knowledge discovery performed by time series databases contains stages such as data preprocessing, feature extraction, transformation, dimensionality reduction, prediction, and rule extraction (Last et al. 2001). Much of the previous research has focused on analysing time series data, aiming to provide relevant results directly to human users, possibly in symbolic form. Although the analysis and modelling of time series for forecasting purposes has been widely studied theoretically, there is a lack of robust enough algorithms for real-life usage and a lack of easy-to-use software tools (Gooijera and Hyndmanb 2006). It has been stated that no single method provides results that alone are enough when diagnosing process operation (Venkatasubramanian et al. 2003a). The major reason for this is that data processed by a single method do not contain all the relevant information and unknown issues influence the end result too much.

One useful way of abstracting time series data to symbolic form is to find episodes from them. These are time intervals and periods that are delimited from each other by significant change points or value outliers in the trends. Numerous methods have been successfully demonstrated to realise this; for example, the Monte Carlo strategy and particle filtering are reported to be suitable methods for the detection of change points in long time series (Chopin 2007) and several other methods have also been reported (Lavielle and Lebarbier 2001; Lowe et al. 1999; Takeuchi and Yamanishi 2006). A concrete utilisation example of detecting change points from time series utilisation has been demonstrated in a case where it was used to detect the influence of human activities on water levels in rivers (Wong et al. 2006).

Gentil (2006) reports the application of trend analysis and the use of symbolic information related to process monitoring. They see that time series of individual process quantities may be classified to form linear segments, which may be further categorised into seven different types [*Increasing, Decreasing, Steady, Positive Step, Negative Step, Increasing/Decreasing Transient, Decreasing/Increasing Transient]*. The use of a simplified version containing only [*Steady, Increasing, Decreasing*] classification is also reported to be useful. The symbolisation is stated to be directly useful for the human operator but also usable for forecasting purposes. Predicting that a current linear segment continues, it is possible to forecast that a certain parameter will exceed the alarm limit in a certain time span. Seppälä and Salmenperä (2005) have also proposed similar usefulness for linearisation.

In conclusion, it can be stated about available methods, on the basis of the reported research and related demonstrations, that with mathematical methods it is possible to find linear periods from time series data. The accuracy is related to the predefined description of the analysed data and parameterisation of the methods used. On the basis of the literature it is relatively safe to state that if less than 100% accuracy in detection is enough then it is possible to gain symbolic information using statistical methods, even with a feasible amount of configuration work. Furthermore, it was found that finding episodes from offline data is easier and more accurate than finding episodes in real time; however, in process-related monitoring applications online operation is required for the timely detection of changes. Nevertheless, every method has its operational restrictions and these should be considered when applying them in order to be able to get feasible results.

3.6 Challenges and technical opportunities in process automation monitoring

The background and general motivation for this research is that functionalities in process automation monitoring should be looked at from a new perspective. On the one hand, there are new challenges related to monitoring in process automation, especially the need for flexibility and delegation, because the complexity is rising and sites are growing in size. These challenges were discussed in Chapter 2. On the other hand, the technological opportunities presented above in this chapter are interesting in the context of monitoring. Synthesising these two research areas together is not a unique idea, and therefore related research activity has been reported. The following part of the text focuses especially on properties such as flexibility, delegation, systems integration, and knowledge handling which are seen as important in process automation monitoring systems.

Flexibility and control over operations is stated to be the main user need related to process automation information processing in this thesis. This means that the system structure should support the flexible organisation of functionalities. In addition, the user should be able to decide how, when, and what information-processing functions are to be used during monitoring. Current automation environments offer numerous tools for specific tasks, but in general the user is not able to integrate and fine-tune their simultaneous operation (Koskinen et al. 2003; Paunonen 1997; Seppälä and Salmenperä 2005; Tommila et al. 2001; Ventä 2005). Techniques for gaining flexibility are available in software engineering, and these have been at least demonstrated in the area of automation. Agent technology has been used to achieve

flexibility when integrating industrial systems (Buse and Wu 2007; Cockburn and Jennings 1995; Mangina et al. 2001; Shen and Norrie 1998; Shen et al. 2006; Wagner 2002) and executing batch process operations (Kuikka 1999). Agents have been used as a technique in flexible process control (Chokshi and McFarlane 2002) and, for example, in realising fault tolerance (Maturana et al. 2005; Seilonen 2006).

It has been proposed that agent-based principles could be used to perform totally distributed process control, in which individual product elements find the necessary processing facilities and materials for each phase to form a complete end product (Chokshi and McFarlane 2008). Demonstration experiments have shown how agents may be used to handle dynamics in electricity production (Buse et al. 2003; Buse and Wu 2007; Kok et al. 2005). In manufacturing the potential of agent systems for gaining flexibility has been researched more extensively and for a longer time period (Barata et al. 2005; Beckstein et al. 1994; Chokshi and McFarlane 2002, Jennings and Bussmann 2003; Marik et al. 2002a; Shen et al. 2006) or structuring has had similarities to agents (Brennan et al. 2002; Vyatkin 2005). However, other base technologies have also been proposed when reaching for flexibility in automation. Lately, especially Service-Oriented Architecture (SOA), designed primarily for business processes, has also been applied to more practical and floor-level industrial applications (Jammes and Smit 2005; SIRENA 2008), and also Web Services (Lastra and Delamer 2006).

Delegation is needed because the amount of data to be processed and monitored when making decisions has grown with the rise in the general complexity of automation systems and the technical aspects of IT development, and as a result a need for more flexible ways to delegate monitoring functions is clearly visible (Koskinen et al. 2003; Seppälä and Salmenperä 2005; Ventä 2005). Technically, there are numerous tools to construct delegation, and agent technology implementing proactive computing principles has been stated to be probably the most interesting one (Maes 1994; Tennenhouse 2000).

General monitoring tasks, which are structured, repetitive, and relatively simple, have been proposed to be well-suited functionalities for AI technologies (Kaplan 1984; Laplante et al. 2007). Agents and AI technologies were already being used for cooperative information gathering over ten years ago (Bayardo et al. 1997; Decker et al. 1995) and also for monitoring supporting activities in industrial settings (Cockburn and Jennings 1995; Ingrand et al. 1992). Lately, industrial activity has gained more interest (Bunch et al. 2004; Georgoudakis et al. 2005; McArthur et al. 2005; Theiss et al. 2007; Wang et al. 2004; Wörn et al. 2002). Furthermore, when the monitoring of equipment is being outsourced and performed physically further away from the production site, then agents may be used to delegate the monitoring task (Pyötsiä 2005; Theiss et al. 2007; Terziyan and Zharko 2004).

Systems integration is important, because not all the information that is needed to make correct decisions on a production site comes from the process automation system (Ventä 2005). Integrating existing industrial systems together utilising agent technology and languages adopted from the Semantic Web has been suggested as a viable approach (Terziyan and Zharko 2004; Yi-chuan et al. 2006), and it has been stated that another promising way to use agents is to integrate them into existing systems in manufacturing environments (Shen et al. 2006). In power generation it has

been demonstrated that agent technology offers flexibility when integrating various distributed condition monitoring modules and thus creating an easy migration path for new applications (McArthur et al. 2005). In addition, agents have been reported to be a suitable technology for the integration of information management, condition monitoring, and control in power systems (Buse and Wu 2007). The integration of existing systems with agents to support monitoring and diagnostic purposes in continuous processes has also been proposed in the ARCHON system (Cockburn and Jennings 1995), by the COMMAS project (Mangina et al. 2001), by the KARMEN project (Bunch et al. 2004), by the MAGIC project (Gentil 2006; Wörn et al. 2002), and in other efforts (Gao and Kokossis 2005; Sayda and Taylor 2007; Vasyutynskyy and Kabitzsh 2004).

As research has proposed agents for integration purposes, SOA is already becoming a popular solution for the integration of industry applications (Jammes and Smit 2005; SIRENA 2008), also in a vertical direction as a tool to normalise and thus integrate factory machinery to a company's business models (Gilart-Iglesias et al. 2006). SOA has also been proposed for monitoring purposes (Seppälä and Salmenperä 2005). As an industrial practice the OPC specifications have been a widely adopted cornerstone within automation and the forthcoming version will utilise both the SOA and Web Services (OPC 2008). Lately, adding semantics to SOA-based infrastructure has been proposed for power market information integration (Huang and Lei 2007), and successful industrial systems integration has also been demonstrated with Web Services (Kalogeras et al. 2006; Lastra and Delamer 2006).

Production environments are information-intensive environments and successfully organising the **knowledge handling** is therefore crucial, and to support efficient operation user interfaces should offer the right information at the right time in the right form (Ventä 2005). Technically, information exchange in automation environments has been largely solved by the available IT solutions (Sauter 2005), but this does not solve the knowledge issues. Challenges in knowledge handling are being dealt with in the software research community, and especially the W3C (2008) is developing technologies in the form of specifications, guidelines, software, and tools supporting interoperability. Within automation, the new OPC Unified Architecture specification will support the use of formal and descriptive data models (OPC 2008), and these may be used to add knowledge handling and process modelling functionalities to services. The use of semantic knowledge has been demonstrated within industrial settings (Georgoudakis et al. 2005; Huang and Lei 2007; Lastra and Delamer 2006; Vyatkin et al. 2005).

Research activities have already been reported that explore new possibilities for knowledge handling and process modelling purposes in automation. Obitko and Marik (2003) have stated that OWL (2008), the latest ontology language proposed by W3C, would be suitable for information exchange in manufacturing enterprises. Semantic models have recently been used in maintenance information processing by Viinikkala et al. (2006) and also for flexible power market information integration by Huang and Lei (2007). However, the latter note that an agreed and shared vocabulary (ontology) for integration is lacking and should be publicly available in the future.

Process models are often used in the monitoring of industrial systems for fault detection and diagnosis purposes, because they offer benefits such as accuracy, reaction speed, and identification (Honkanen 2004; Isermann 2004). In the future the modelling of the whole production environment and its efficiency will become more important, because of requirements originating from business and environmental perspectives (Tuomaala 2007). Nevertheless, models and rules describing the operation of the system need an estimation of parameters if they are to be useful and this is stated to be problematic in environments where change is present and therefore control over decision-making should be left to the users (Laplante et al. 2007).

Table 2 shows a collection of challenges originating from users trying to cope with monitoring tasks in the process automation area and the opportunities that the above mentioned new technical solutions are reported to offer. The table also shows selected references that discuss these aspects and provide further information about the subject.

Table 2 – Collection of how different technical solutions have been used to respond to challenges.

Challenges for the system originating from business change	Suggested technical solutions	Important references
Flexibility	Agents, SOA, Web Services, Semantic Web tools	Buse and Wu 2007; Chokshi and McFarlane 2002; Wagner 2002; Jennings and Bussmann 2003; Jammes and Smit 2005; Shen et al. 2006
Delegation	Agents	Maes 1994; Tennenhouse 2000; Georgoudakis et al. 2005; McArthur et al. 2005; Wagner 2002
System integration	Agents, SOA, Web Services, Semantic Web tools	Bunch et al. 2004; Buse and Wu 2007; Cockburn and Jennings 1995; Huang and Lei 2007; Kalogeras et al. 2006; Mangina et al. 2001; Wörn et al. 2002
Knowledge handling	Semantic Web tools	Georgoudakis et al. 2005; Huang and Lei 2007; Lastra and Delamer 2006; Obitko and Marik 2003; Viinikkala et al. 2006; Vyatkin et al. 2005

On the basis of the reviewed literature it is possible to conclude that agent technology seems to offer most of the properties that are needed to construct a monitoring system. Compared to SOA and Web Services, which have already been significantly adopted within industry practice, agents seem to offer better support for delegation. Although applying agent technology to practical industrial applications is still rather rare, it has been seen as an important research direction. A steady development trend is visible in the series of HoloMAS conferences (see Marik et al. 2003, 2005, 2007) that were first initiated as a workshop (HoloMAS 2000, 2001, 2002; Marik et al. 2002b) but have had growing attention since.

Previous research has studied monitoring systems realised with agent technology, possibly utilising the Semantic Web, in environments bearing similarities to process automation. One of the earliest was the ARCHON system (Cockburn and Jennings 1995), which used agents to modularise and integrate pre-existing expert systems in an electric distribution network. The aim with ARCHON was to use agents to

organise the sharing of partial results together for combined fault diagnosis. In addition to well-documented experiments, the ARCHON project generated guidelines for the systematic development of monitoring systems within industrial applications, and demonstrated the use of AI planning. ARCHON-like structures have also been suggested for the monitoring of automation systems in general (Vasyutynskyy and Kabitzsh 2004) and for enabling the G2 expert system to be used in the petroleum industry (Sayda and Taylor 2007).

The COMMAS project, which has a similar background to ARCHON, uses agents to flexibly organise condition monitoring operations in power plants (Mangina et al. 2001). The agents have different roles (data abstraction, data processing, analysis, and administrator) and are organised hierarchically in multiple layers. Communication in COMMAS is based on the FIPA standard, although XML is used as the message content language for practical reasons (McArthur el al. 2005). Defining agent roles on the basis of their diagnostic tasks is also used in the MAGIC project (Wörn et al. 2002), which also uses FIPA standard interaction protocols in communication. In both COMMAS (McArthur et al. 2005) and MAGIC (Gentil 2006) the data analysis agents extract features from time series data and forward this abstracted information for further usage.

The studies introduced in Buse et al. (2003) take the ideas introduced in ARCHON further. Their system has multiple layers and agents are thought to have various roles, motivated by functional task division (database, document, ontology, device, plant, and user interface agents). In addition, the idea of planning is concretised in the design by defining agents to use a BDI model for deliberation. However, BDI was not used in the implementation, because of the amount of time required to implement it and because the functionality did not require it. Furthermore, the FIPA standard is used in communication, and even an FIPA-specified content language is used. In addition, the system uses an ontology defined with UML, e.g. to describe different items found in the system (Buse and Wu 2007).

The KARMEN project (Bunch et al. 2004; Bunch et al. 2005) utilises agents to distribute monitoring tasks for physical processes in a similar way to the MAGIC project. However, the focus in KARMEN is on notification issues, e.g. being able to relay each notification to the appropriate person. The Semantic Web ontology language OWL (OWL 2008) is used to model aspects related to notifications, such as the criticality of the notification or the person responsible, and this same model is used to control the notification process. Other researchers (Georgoudakis et al. 2005) have also used ontologies and agents to organise monitoring-related activities, and suggest the use of languages originating from the Semantic Web in inter-agent communication. Furthermore, Mathieson et al. (2004) have successfully experimented with BDI-based agents in relation to a meteorological alerting system.

As a conclusion to the consideration of the use of agents for monitoring functions within industrial settings, the following may be stated: agents can be used to distribute the monitoring activities and construct hierarchical structures that are suitable for the task. A BDI model and Semantic Web tools have both been proposed as being useful in monitoring. However, demonstrations utilising both at the same time seem to be missing.

3.7 Conclusions about the available agent and semantic web technologies

Computing power has been increasing remarkably and provides many possibilities in system development. Although it is not clear what the best choice for utilising the extra capabilities will be, raising the level of abstraction in system development seems to be appealing. Previously the focus was very much on constructing effective algorithms and structures that optimise the exploitation of the capabilities of the hardware. However, if the system is complex and applications must function in a dynamically changing environment, then the capabilities of the approach used should ease the work of system engineers. Agent technology is proposed as one possibility capable of answering this general requirement (Jennings 1999; Odell 2002).

Agent technology itself is a rather general software engineering methodology suitable for system-level design. It promotes the system to be constructed from abstract entities that are autonomous and communicate by passing messages. There is an IEEE standard available (FIPA 2008) that defines agent communication and defines negotiation principles on a general level. Numerous software frameworks for constructing agents are available and the quality of these is on such a level that these could be used for selected industrial applications. Systematic methodologies and tools for developing BDI model-based rational agents are available (Padgham and Winikoff 2004). However, agents have the limitations of general AI (Wooldridge and Jennings 1999), and although AI seemed promising in the past, adaptive systems with intelligent and highly autonomous features remain something of a fantasy (Fischer and Merritt 2003; Hendler 2006). It has been proposed that decisions on what features to include should be made in the context of the motivating problem domain and that unrealistic assumptions should be replaced with practically justifiable ones (Fischer and Merritt 2003).

In a technological sense, semantic technologies are gaining a lot of interest because they offer potential to access the vast amount of information provided by the internet. Standards for representation issues on a syntax level and metadata definition languages for defining semantics are already available. However, the methods for ontology engineering are still under development and no widely adopted use of semantic information has been seen in industrial applications. Service-based system structures are gaining strong interest within business applications and frameworks for this are starting to become mature. Nevertheless, current approaches require the user to define the procedure that describes how services are utilised, and do not promote automatic service composition (Lassila 2007).

Interest in combining service-based operation and semantic information approaches is already visible, and the idea of Semantic Web Services has been introduced. Numerous tools, systems, frameworks, and methodologies are available in the research community, but none of them are usable as such for process automation monitoring purposes. One specific example combining Semantic Web tools and agent principles together is the Nuin agent platform (NUIN 2008). Although Nuin could have been useful in the scope of this thesis, the development project of Nuin was stopped before the platform reached sufficient maturity. Therefore, as there are no off-the-shelf systems available, the system has to be constructed from selected, suitable, and individual parts that are available.

In general it has been stated that intelligent systems aiming to support humans should be easily configurable because in many situations the end user knows the situation and meaningful changes the best. Furthermore, systems aiding humans should perform laborious filtering, searches, and monitoring, but not too much else (Laplante et al. 2007). Systems should offer flexibility in configuration so that the user may personalise the system and select the level of automation that suits each task best (Schiaffino 2004). Related to the monitoring of automation systems, an appropriate solution seems to be a flexible user-configurable system that has models for relatively static aspects (process ontologies covering structural aspects of the process environment and base vocabulary) to facilitate communication and enable agents and other active players to perform the functions that have potential for automation (e.g. with user-defined rules for inference and filtering, etc.).

4 The new agent system architecture for process monitoring

4.1 Introduction

The aim of this chapter is to present an agent architecture design that facilitates the technological opportunities presented in the previous two chapters and responds to process automation business challenges. On the architectural level the design choices focus on selecting the most suitable overall structure, including the selection of an appropriate agentifying level and the enabling of connections to other industrial systems.

Categorising the currently visible development trend in process automation from the structural perspective can be done by looking it from two distinct directions; bottom-up, describing the rise of information-processing capabilities, and top-down, describing the advances in the management capabilities of the system structures.

Bottom-up development trends include: in the future, nearly every device connected to process automation will diagnose itself and possibly monitor the health of nearby process areas. Device vendors will implement more utilities into their devices as developments offer more memory and processing power, e.g. device-specific data mining may be introduced. Furthermore, vendors are already supporting external programs to be run on their devices, e.g. simple logic controls are implemented within electrical drives.

Top-down development trends include: the complexity of processes will grow because of increased integration and more measured information becoming available. Models describing a variety of more or less general issues about processes will become available, e.g. tighter integration between process design and implementation will make descriptive models available. In addition, methodologies for distributed software development will evolve and provide more sophisticated approaches to the handling of modularised design, e.g. agent technology. The tools and techniques for handling integration issues and interpretation models are also maturing, e.g. Semantic Web tools may be used to integrate and query data available in heterogeneous data sources.

Although process automation is likely to adopt many of the new technologies, process control will probably be organised in a hierarchical structure in the near future (Jämsä-Jounela 2007). The hierarchical organisation of process control offers unbeatable reliability, even though distributed approaches have also been demonstrated to be possible in process control (Chokshi 2005; Seilonen 2006). Regardless of whether or not process control becomes more distributed, the monitoring functionalities may be organised in a distributed manner. Distribution approaches should also be preferred that are able to exploit all the possibilities that new embedded computing offers, e.g. to overcome the limitations of the current approaches (discussed in Chapter 2.5).

This chapter presents an architectural design that illustrates how to organise the whole system utilising agent technology to support flexible monitoring in process automation setups. The architecture illustrates a structure which aims to help developers to integrate various isolated functionalities together and further facilitates the combination and summing up of partial results. In addition, the overall structure aims to be understandable and feasible to implement in process automation environments. Furthermore, the structure should promote cumulative development and make future additions possible.

The first part of the design is the setting of the requirements for a flexible process monitoring system. The requirements are represented in the form of desired system properties in Section 4.2. Then the rest of this chapter presents the design of an agent system architecture for process automation monitoring point by point, and in every section the design choices are explained. Furthermore, as the design of the system and its functions is a continuous and evolutionary process it should be noted that this thesis documents the current status of this work.

4.2 Desired system properties

The target is a system that would provide user-controllable, flexible, and easy enough access to all-available information related to process monitoring tasks within an industrial production site. Although it is end user-configurable to a large extent, at the same time it should exploit the technological benefits coming from novel software engineering research.

The discussion in previous chapters has yielded properties that an architecture for flexible user-controllable and configurable process monitoring that produces operationally integrated information services should have. Below these properties are clustered into six groups. The first five were introduced at the end of Chapter 2, and here these are discussed from the system development viewpoint. The final property, general properties, promotes design issues for a maintainable system structure.

P1 - Flexibility: the system should support *context sensitivity and adaptive operation*, to enable the user to control how the system behaves in different situations. The level of information also needs to be controllable; in a steady situation abstract reports are enough, unlike in an abnormal situation, where specific details are needed. Furthermore, the monitoring functionality requires adaptive, asynchronous, and frequent checking of things that in their own way show that everything is as it should be. The system should support the *active operation* principle, to enable the user to automate as much of the monitoring operation as possible.

P2 - Delegation: the system should enable the user to use *delegation* for most of the routine work and thus release the user to concentrate on higher-level decision-making. This requires the system to be somewhat intentional and rational and should also enable short procedures to be configured for it. The system should be able to process multiple tasks simultaneously and in parallel with *distribution*. Distributed operation is required to be able to utilise promised future improvements in the device-level diagnosis functionalities.

P3 - Integration: the system should provide *integrated information access* to make possible one-point access to all the process-related information that is needed when validating the correct operation of the process and devices. Furthermore, *merging to current systems* with an architecture connecting to current structures is required because the life cycles of industrial control systems are long in process automation. Support connections to pre-existing information and control-level systems in process automation are required.

P4 - Knowledge handling: the system should enable *unification* of terms used for the essential data model items and thus facilitate *linking* of (partial) results provided by a variety of tools, e.g. being able to combine monitoring results together. In addition, the use of both the functional and physical decomposition of underlying processes should be supported by the system. The system should be able to *operate with incomplete information* and still provide results, although partial, even when all the requested information is not available. Support for the all-you-can-find principle is preferred.

P5 - Data processing: the system should be capable of performing data processing suitable for process automation, especially *extracting* and *abstracting* relevant information from time series format data.

In addition to process automation-motivated properties (P1-P5), the system should have the following **general properties**: the system should be *modularly structured* for easier maintainability, e.g. to support flexible updating in the future. Separating the control of operation and decision-making is preferred. Modularity aspects and operating principles should be intuitive and use structures from the application area and its functions as much as possible. *Intuitive operation* is also preferred, because the end user should be able to perform limited system operation configuration, as in the process automation environment the end user has the most timely information about the current situation.

Although the system will be situated in a process environment, where stringent real-time constraints are typically important, real-time issues are not included in the properties. This is because the monitoring system mainly provides supportive information, where the time constraints are thought to be soft. Information that has been asked for should be returned as fast as possible but reasonable delays are not seen as problematic and processing can also be performed in the background. This is different compared to systems that automatically make corrective actions, as discussed, for example, by Ingrand et al. (1992) and Laffey et al. (1988). In addition, what is missing as a stand-alone property is interaction with the end user. This thesis treats system design as a structural and technical problem, and the design of the exact user interface and interaction for monitoring tasks are seen as the subject of other research.

4.3 Agent augmentation

To be able to use agents in a process automation environment, a suitable overall structure needs to be defined. As automation setups have long lifetimes (e.g. 20-30 years) and the monitoring tasks are supportive functionalities, the most important structural requirement for an agent system is to be able to add it to current industrial setups without requiring modifications to the systems' current structure. Figure 10

illustrates the specified agent-augmented approach in which the agent system is situated in the middle of various interconnected systems that already exist in the process automation environments. With this structural setup the agents have the potential to provide additional functionalities to users, while at the same time the pre-existing systems and their connections are left untouched. The system is primarily thought of as being suitable for use by humans but the monitoring functions and their results might be used by other systems and applications as well.

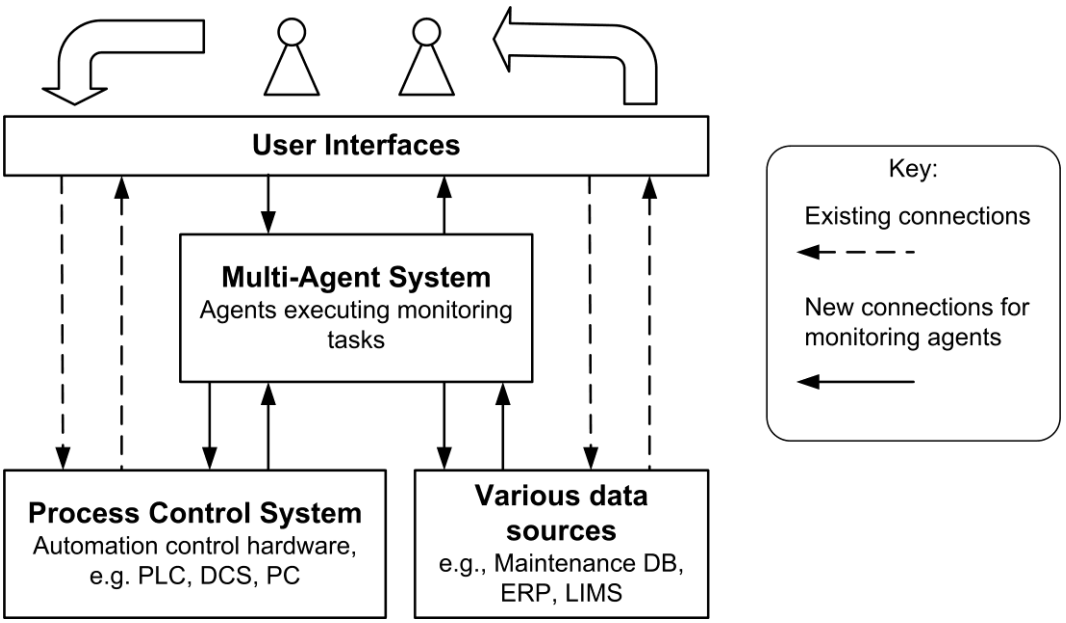

Figure 10 - Overall architectural structure illustrating agent augmentation.

Keeping the agents as a clearly separate system eases the adoption of the agents, as the system may be installed as a stand-alone module. This enables available agent software frameworks to be used and makes the realisation relatively straightforward. This augmentation approach was selected instead of trying to incorporate the agent functionalities into current process automation setups, leaving the reliable real-time control structures and functions intact. However, if in the future the automation environment enables various tasks and applications to be performed in a distributed manner, then the agent-based monitoring functions can also be distributed. The overall system structure, illustrated in Figure 10, is designed to be static and it is not thought to be changing at the time of performance. Furthermore, the augmentation approach keeps the monitoring system compatible with the previous research of applying agent technology to enhance process automation system properties in control operations (Seilonen 2006).

4.4 Members of the agent society

Inside the agent system the number of agents and their setup changes dynamically, according to the currently active monitoring tasks. The agent hierarchy is intended to be suitable for process automation monitoring tasks. The responsibilities of each agent operating in the system are defined beforehand and in this architecture these responsibilities are specified as roles in which each agent operates. We propose that this role setting eases the design process as there are guidelines for selecting the place for every function. Furthermore, guidelines also help when defining an organisational hierarchy for agents and roles are seen as a way to cluster functionalities into

meaningful combinations, as the role of an agent outlines its area of responsibility, objectives, and behaviour. Figure 11 illustrates a general example of agent organisation showing all five different roles in which the agents may be set to operate.

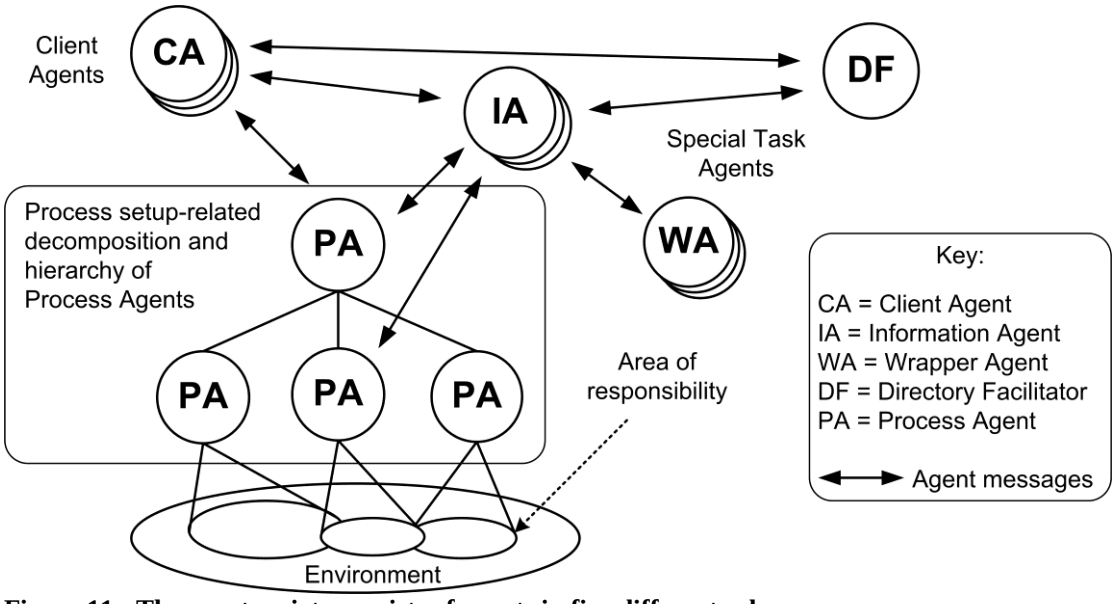

Figure 11 - The agent society consists of agents in five different roles.

From the user's point of view, the *Client Agent* is probably the most important agent type because it provides a gateway to access and control the monitoring services. The task of the *Client Agent* is: 1) to support the tasks of the user by providing information retrieval and monitoring services; 2) to translate agent communication into forms that are user-understandable, and 3) to maintain a conversation with the users in accordance with the current agent task. Although the system is mainly thought to be suitable for use by human users, external software systems might also use these monitoring services via the *Client Agent*. Existing user interfaces may be used if the agent functionalities can be embedded into those, but, when needed, specialised user interfaces should be produced. Normally, these *Client agents* persist for as long as each user is using this system.

Information agents are service providers that have a less permanent lifespan in the system, as they are typically instantiated for the performance of a specific information task. Their task is to carry out information retrieval and processing tasks that require the migration of data from a number of service providers. The responsibilities of this agent role are to: 1) decompose information retrieval and monitoring tasks into queries and subtasks for appropriate agents; 2) filter, combine, and format the collected data using various methods, and 3) monitor for phenomena dependent on the distributed initial data. *Information agents* may be seen as specialised consultants that are invited to perform some particular function and they may be dedicated to these functions, e.g. in the same way as an individual service engineer may specialise in a certain device type at a production site. These agents are capable of decomposing the process environment and its operations from multiple different viewpoints, e.g. functional and physical, at the same time and thus providing users with a view of information that is not readily available in any pre-existing system.

A *Process Agent* represents a section of the physical plant equipment, and they are thus more permanent than the previous agent roles. *Process Agents* are structured hierarchically, with low-level agents representing single devices such as pumps or valves. On a higher level, process agents account for plant (sub)processes, and on the highest level for entire processes or plants. These agents maintain a direct connection to the hardware and employ various sets of methods in order to: 1) collect and refine information about the operational state of the equipment; 2) attempt to discover deficiencies in the performance of the equipment, and 3) monitor for changes in the desired information (e.g. measurements) both by default and on request. The idea of decomposing the physical process into hierarchical subprocesses and setting *Process Agents* to be responsible for these originates from previous research exploring the possibilities of an agent approach to process control functionalities (Seilonen 2006).

The role of *Wrapper Agents* is to be responsible for information sources that are already available in the process automation environment. Their tasks include: 1) maintaining a data connection to the information source; 2) translating data from different formats to the ontology-based representation understood by the agent society, and 3) filtering and monitoring for changes or updates in the information stored in the data source on request. The use of wrappers supports the evolutionary development of the whole system as only a new wrapper is needed when a new information source is added. Typically, there is thought to be one permanent *Wrapper Agent* for each external data source, but this is not a limit that is intrinsic to the design.

The final agent role in the system is that of the *Directory Facilitator*, which provides a yellow pages service, advertising and seeking services. This role is defined in the FIPA standard, which requires an agent system to provide a directory service so as to be compliant with the standard (FIPA 2008). The directory service facilitates dynamic task (re)configuration and plug-and-play interaction among agents, and it is typically a built-in feature in the actual implementation of agent systems.

The presented role division supports general intuitiveness in the design process, as these roles are related to the roles in which humans work in industrial contexts. Thinking of the presented role definitions from the software engineering perspective helps in keeping the agents relatively small in terms of their functionalities and thus helps in achieving low coupling and high cohesion, which are typically stated to be important design objectives in modularised software systems. The presented roles may also be preserved if the whole system is to be physically distributed all around the process environment. In addition, the described roles are a realisation of the future vision presented in Pirttioja (2002) about different types of agents operating all around the process automation environment.

A practically identical type of agent structuring is proposed by Buse and Wu (2007) for information management and the condition monitoring of power systems, but they name their agent roles differently and they have included mobile and document agents in their system. Similar ideas of composing physical processes into separate elements have been proposed by others too (Buse et al. 2003; Buse and Wu 2007; Chokshi and McFarlane 2002; Wagner 2002). The concept of an information agent providing services in relation to automation functionalities has been suggested by Wagner (2002). In addition, the wrapping idea has been utilised in information-related

applications in power systems (Buse et al. 2003; Cockburn and Jennings 1995; McArthur et al. 2005).

4.5 Agent interactions and service directory

Flexibility and adaptation in agent systems are the result of both the autonomous operation of individual agents and also the correctly organised loose coupling of agents. This coupling is based on various communication methods, called interaction protocols within agent technology, and they are standardised by the FIPA organisation (FIPA 2008). Although other technologies also offer flexible ways to communicate in a multi-entity environment, the FIPA standard offers strictly defined and stable guidelines for realising adaptive communication in distributed environments.

Analysing the task communication requirements and selecting the most suitable interaction protocol for each task is important because correctly selected communication principles support the functionality. For monitoring and information access purposes, the most interesting protocols that agent standardisation offers are the *subscribe* and *request-when* interaction protocols. With these interaction protocols the actual monitoring activity may be delegated and the requester is released from the resource-consuming polling of data and testing if something has been changed. For example, in the *subscribe* interaction protocol the client is notified every time the referenced object changes until the subscription is cancelled. The *request-when* protocol may also be used for delegation but with it the requested action is performed once. For information access purposes, the *query* interaction protocol can be used.

Although the FIPA standardised interaction protocols were seen as being valuable, it was decided not to use the FIPA-specified content languages in the communication. This was partly because insufficient support is offered by the available software tools. The primary reason not to use the FIPA standardised agent content languages was the unclear situation related to other technologies under heavy development, e.g. Web Services (Chapter 3.3.1) and the Semantic Web (Chapter 3.4), and this has also been noted by the FIPA organisation (Greenwood et al. 2007).

Furthermore, an important source of flexibility and adaptation is the service directory, provided by the *Directory Facilitator* agent introduced in the previous chapter. With the service directory information sources are searched through when they are needed, not at the time of design, which results in the adaptive and dynamic binding of resources. In addition to guaranteed up-to-date information, this ensures that new information sources are immediately available when they are connected to the system.

4.6 Goal-based operation and shared ontology

To succeed in monitoring tasks both reactive and deliberative operation is needed. From the basic agent types, discussed in Chapter 3.2.2, the deliberative and goal-based operation realised with the BDI model seemed to be most suitable for monitoring purposes. The BDI model suits the adaptive and repetitive (goal-pursuing) type of operation that is needed for controlling monitoring tasks. However, reactivity is needed so as to be able to respond in a timely manner to changes occurring in the process automation environment. It was decided to support both types with a construction in which the deliberative part is responsible for configuring and thus controlling the operation of the reactive parts.

In the BDI model the goals (desires in the general model) are used to describe information-processing and process monitoring task objectives. The general idea is to imitate how humans give instructions to each other and provide a more intuitive operating principle for the monitoring system. Furthermore, passing goals between agents enables receiver agents to decide locally how and when to operate, e.g. how to acquire the requested information. This results in a more flexible setup than the remote procedure calls that many distribution frameworks are based on, e.g. Java. The BDI model supports flexibility in the form of context adaptation. With it it is possible to describe the preconditions for each individual information-processing functionality that defines in which situations it is capable of producing the outcomes that it is designed to produce. The use of goals promotes distribution and task delegation as agents may request other agents to reach goals, e.g. in a case where the agent itself does not have the abilities to fulfil the task itself.

In addition to a principle that controls agent operation, agents need a common and shared terminology for communication purposes. In general, these are defined as ontologies, which are explicit conceptualisations of a certain domain of knowledge. Common agreement on this knowledge is needed for agents to be able to exchange meaningful information between them. In this agent system this is realised with an ontology that is shared knowledge available to every agent in the system. In order to avoid the n^2 translation problem, every agent is required to supply its information using this vocabulary. For example, the *Wrapper Agents* are responsible for translating their responses so that the requester understands the presentation format and is able to utilise the information.

In addition to shared terminology, the ontology is used to define the relationships between the concepts used. This can also be used for reasoning purposes inside an agent. For example, the possibilities and readiness of the Semantic Web-related tools and standards (see Chapter 3.4.2) are attractive. Unfortunately, currently there are no standardised or otherwise publicly available ontologies for process automation, and especially not for monitoring functionalities. Nevertheless, in this research the decision was taken to develop a demonstrative ontology for the experiments so as to be able to test the idea. The utilisation of ontologies for representing the structure, events, and behaviour of industrial processes is described in more detail in Pakonen et al. (2007).

4.7 Summary of architecture decisions

The agent system architecture presented here aims to provide a basis for building flexible and configurable monitoring services for users in process automation. The architecture is a collection of design decisions made to adapt agent technology to monitoring tasks. Table 3 summarises how, it is argued, the design decisions relate to the desired system properties, which were set for the architecture in Chapter 4.2.

Table 3 – Summary of architectural elements and their relation to desired system properties (the mark X indicating that the architectural decision supports the property at least partly).

	P1: Flexibility	P2: Delegation	P3: Integration	P4: Knowledge handling	P5: Data Processing
Agent augmentation	X		X		
Agent roles	X	X	X		X
Agent interactions	X	X	X		
Goal-based operation	X	X			
Shared ontology			X	X	X

Realising agents with augmentation was selected primarily because of the technical limitations of the current automation environment. As the augmentation enables the system to be used with current structures, it supports integration and flexibility. The use of agent roles gives guidelines to the system designer and helps in structuring the operation of the system, thus supporting flexibility and delegation. Wrapper agents in particular may be seen to support integration and data processing aspects. However, the author expects that the exact role division presented will be subject to change if new functionalities are developed. The use of FIPA (2008) standardised agent interactions is one of the key features offered by agent technology and it naturally supports flexibility and delegation. Furthermore, negotiations and directory services facilitate the integration of systems.

The goal-based operating principle was selected mainly for delegation and flexibility purposes, but also because it has been stated to be suitable for building applications that are used by humans (Dickinson 2006). Additionally, a systematic design methodology is available for BDI model-type agents (Padgham and Winikoff 2004). However, although the BDI model seems to match the requirements set by monitoring tasks in automation, no specific examples reporting its use were found. This results in a risk as there are no good guidelines showing how to apply BDI model-based operations to monitoring in process automation. Shared ontology-based communication is considered to ease the integration of data sources using heterogeneous data formats. Furthermore, ontology-based data models naturally support knowledge-handling issues, and the possibility of describing object relations in ontologies is seen as potentially beneficial to data-processing functions.

5 Layered agent design and its functionalities

5.1 Introduction

The system architecture specified in the previous chapter makes design decisions on an overall level and these choices need refinement and more details to be specified before they can be implemented as an actual software system. In addition, the system architecture design mostly explains the multi-entity aspects of the system, e.g. what kinds of players there are in the system, and does not explain how the individual agent has been constructed or how the operation of the agent is to be realised.

The aim with the internal design specification is to support both deliberative and reactive agent operating principles in a suitable way for monitoring tasks in process automation. Furthermore, the specification should give guidelines to the designer as to how to divide the monitoring operation and related data processing into different software modules. Related to data processing, a vertically layered architecture seemed to describe most naturally the abstraction used inside an agent. The layered design illustrated in this chapter is propositional.

5.2 Internal operation of agents

The developed internal architecture specification, illustrated in Figure 12, is a modified and mixed version of a vertically layered architecture and the generally known BDI model. Both of these were discussed in Chapter 3.2.2. The *Task management*, *Belief*, and *Plans* modules originate from the BDI model, and these modules are used to control the monitoring operation performed by the underlying layers. The *Data access and composition*, *Creation of symbolic data*, and *Making inferences* modules together realise the vertically layered architecture organising the data processing for monitoring tasks.

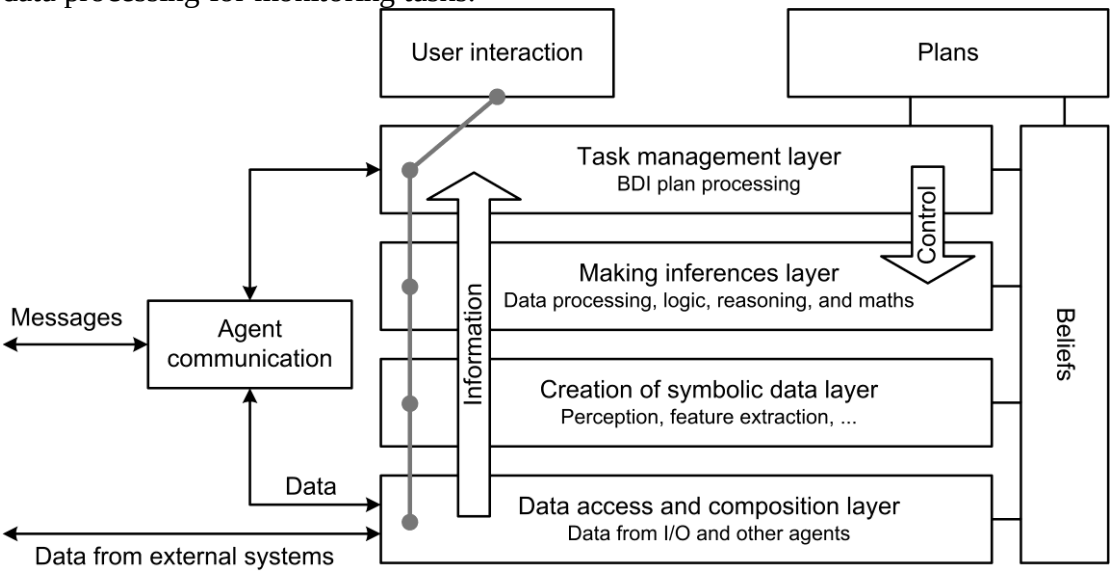

Figure 12 - Internal structure of agents, illustrating connections between different layers. The figure also shows how user interaction accesses information and data through all the layers.

The *User interaction* module operates as a link between the user and the agent system, and the *Agent communication* module presents the connection to the message transport mechanism used for inter-agent communication.

The data processing is divided into three distinct modules. The *Data access and composition* modules are used to access and combine data fetched from several sources with appropriate IO drivers. The *Creation of symbolic data* modules are responsible for processing raw data (usually numerical) into forms (typically symbolic) that are more usable when drawing conclusions in the *Making inferences* module. All these modules contributing to data processing are of a reactive type.

The whole monitoring operation is controlled by the *Task management* layer and its operation is based on the configured *Plans*. The *Belief* module is a common data store, which is also available to the data-processing modules. These three modules together are the source of rationality in terms of what monitoring functions and modules are to be selected for each task.

5.2.1 Data access and combination

By *Data access and combination* data is collected from separate external data sources and transformed into a suitable form for the rest of the information-processing layers in the architecture. This module is used as an adapter or a driver to connect to the data storages available in the automation environment, e.g. pre-existing data bases or external systems. When connected to data sources offering data in mixed format and syntax this layer's task is to convert data into a united and understandable form. This type of operation is especially useful with *Wrapper agents* when accessing information that is stored in legacy databases; see the discussion in Chapter 4.4.

The data-accessing modules are in close connection with the creation of symbolic data modules. In data storages that provide symbolic and event-based information naturally, the data symbolisation process may be bypassed, and the *Data access and combination* layer is used directly by the upper layers. However, in cases where feature extraction or other data pre-processing is needed the two lowest layers work in close connection.

5.2.2 Creation of symbolic process data

The purpose of *Creation of symbolic data* is to transform the selected part of the numerical data into symbolic form so that it is usable in the upper layers of the architecture. Its purpose is to generate information expressing significant and meaningful details, events, and changes in the data and mediate it for further processing, e.g. for modules in the *Making inferences* layer. In our system the sources for numerical data are mainly various types of perceptions from the physical world, e.g. values of direct physical measurements such as temperature or values that more complex instruments such as chemical analysers provide. As these real physical measurements always have more or less noise attached, one important purpose of this layer is to filter the noise so that the information is more easily exploitable in the other modules.

The symbolisation is needed to provide meaningful information for inference-making. Within process automation-related cases this is typically event-type information extracted from continuous time series data. In this research the symbolisation is specified as a separate module inside an agent, but in the future this feature extraction may be part of the basic features that lower-level devices provide, e.g. as discussed in

Kalogeras et al. (2006), Seppälä and Salmenperä (2005), and Ventä (2005). See the discussion in Chapter 3.5 for examples of trend analyses providing symbolic results.

5.2.3 Making inferences

In the *Making inferences* layer observations are made based on data fetched from multiple sources and offered for use by the underlying layers of the architecture. The idea is to combine and process data so that new deductions are generated. Basically, the human operator configuring the system is the source of the rules and mechanisms used for inference-making. With these modules the idea is to provide controllable and flexible information processing that operates on the basis of user configuration and outputs the conclusions for further exploration.

Normally, the modules in this layer use data in symbolic form, but when needed the symbolisation layer is bypassed and they can access unprocessed numerical data directly from the *Data access and combination* layer. Inference-making generates new information in numerous ways when data available from the monitored process are processed utilising various types of rules. Potential sources of rules are user-defined relations between process values, device type definitions, and data-mining tools.

Several concrete examples of inference-making are given below and are illustrated in the experiment part of this thesis:

- Constraints for checking the validity of process values of numerical data type are used in *Experiment 1 - Temporal monitoring* (Chapter 6.3 and especially 6.3.3). In this experiment the symbolisation process is bypassed and the inference-making is based on unprocessed data fetched from the process. In this experiment the user defines rules that define acceptable value ranges and the relations of multiple process quantities. In this case *Making inferences* automates the checking of the validity of the user-set constraints.
- In *Experiment 2 - Search of process events* inference-making is based on rules that refer to items in formal process models. Utilising rules defining item relations to instance data fetched from data sources, the connections between process events and particular time series data may be provided to the user. See Chapter 6.4.3 for detailed information about the use of data models. In this case *Making inferences* automates the linking of partial information.
- Detection of process "fever" is performed by monitoring the timely rate of process change events in *Experiment 3 – Change point occurrence monitoring* (Chapter 6.5.3). Here the inference-making is responsible for checking if the event rate exceeds the user-set limit in a specified time span. In this case the *Making inferences* layer automates the monitoring functionality.
- Observation of user-defined event sequences is realised in *Experiment 4 – Change point pattern monitoring* (Chapter 6.6.3). In this case the inference engine is responsible for detecting user-specified sequences of process change events. In this case the *Making inferences* layer automates the pattern monitoring on behalf of a human user.

As can be seen above, in all of these experiments the human user is the source of the deduction rules that are used when making inferences. This approach is promoted in this thesis, as the objective is to delegate repeated and relatively easy checking tasks

to a machine but still leaving full control in the hands of the user. This is currently seen as the development direction with the most potential. However, in the future it might be useful to study other possibilities for producing the deduction rules; for example, it might be useful to explore the potential of utilising various machine learning capabilities.

5.2.4 Task management and plans

The purpose of the *Task management* modules, together with *Plans* and *Beliefs*, is to control the operation of underlying data-processing modules. These controlling modules realise the flexible and rational operation of an agent on the basis of user configuration. In our approach the task management is operated according to the BDI agent model (Rao and Georgeff 1995), in which the interpreter performs the monitoring operations described in *Plans*. These modules control agent operation and take into account the current situation in the process and available data. *Task management* can run other modules in parallel.

In *Plans* the user has configured the way in which the operations provided by the underlying modules (*Data access and combination*, *Creation of symbolic data*, and *Making inferences*) are used in combination to reach the set objective. Within every plan the restrictions of this plan should also be defined, and possibly some description of the quality of the actions it takes and outcomes it produces. If there are multiple plans that may be used to reach the same objectives, the system should have a means for selecting the most appropriate one, e.g. in terms of the time used and the accuracy of the processing. More comprehensive discussion about using goals for information-processing in process automation environments may be found in Pirttioja et al. (2004) and Pirttioja et al. (2005).

5.2.5 User interaction

The *User interaction* layer aims to provide the user with a means to control and supervise the operation of the whole agent system. The user interacts mainly with the most abstract information-processing layers, setting the overall goals in the *Task management* layer and clarifying issues available in the *Making inferences* layer. But if and when it is necessary, there is no restriction in the agent design on also accessing data from the lower-level layers. In practice, the user interaction is built with task-oriented (graphical) user interfaces that are each built especially for the function at hand. The conducted experiments 1 and 2 (Chapter 6) give examples of how the user interaction might be realised in practice with the agent system.

In the first experiment, *Temporal monitoring with constraints*, the user interaction is dialogue-based. There the user first configures the monitoring task with a window designed for that purpose. Then the system performs the monitoring task silently from the user point of view, but the user is able to cancel the monitoring if needed, just by referring to the name of the monitoring task that was given to it when it started. When the monitoring functionality detects something that needs to be notified to the user, then a pop-up message appears showing the monitoring task name, the monitoring definition, and the actual values of the measurements that broke the conditions. In this case the user is provided with data directly from the lowest level of the architecture, because the information is relevant for the user when making decisions. But the data are shown to the user only when they contain the values that are of interest to the user.

In the second experiment, *Search of process events*, the system tries to offer a similar look-and-feel to the user to that provided by an ordinary web search engine. First, the user is provided with an interface used to set the search parameters, and the information shown to the user is based on the general, and practically static, knowledge about the current setup at the production site, such as process areas, device types, and available data types. During the actual search process the user is not able to control the operation. When the information being searched for is found the resulting data structure content is shown to the user with a graphical interface built for this specific case. In this case the user interface represents data that are stored in external databases without modifying them but the search process has filtered the items so that the user sees only the items that have interesting relations (user-specified) with other data items.

5.3 Summary of the functionalities

The objective of the presented modularised and layered internal design of agents is to give guidelines for the designer and thus ease the realisation and implementation of the desired functions. Table 4 summarises how the functional layers support the desired system properties that were set in Chapter 4.2. The table does not show *User interaction* as an individual layer, because it is seen as an external function regarding the actual information processing.

Table 4 – Summary of functional layers and their relation to desired system properties (the mark X indicating that the layer supports the property at least partly).

	P1:Flexibility	P2: Delegation	P3: Integration	P4: Knowledge handling	P5: Data processing
Task management	X	X		X	
Making inferences	X		X	X	X
Creation of symbolic data				X	X
Data access and combination			X		X

Task management and plans contribute to the desired properties in terms of providing flexibility and delegation, because with goal-based operation it is possible to construct both of these. Furthermore, knowledge handling is supported by the task management module as it controls what data-processing modules are used for each task. Making inferences naturally supports the knowledge-handling and data-processing properties. Furthermore, the inference modules provide aid in gaining flexibility and help in system integration because it can be used as an adapter, resulting in unified results from various inputs. The creation of a symbolic data module mainly contributes to the knowledge-handling and data-processing properties. As the data access and combination modules operate as drivers used to connect external systems, these modules are seen as supporting the integration property and also contributing to the data-processing properties.

5.4 Realised system for experiments

For the demonstration experiments discussed in the next chapter, an implementation capable of realising the specified architectural and layered agent designs was made. The system was constructed as a combination of multiple purpose-selected Java-based software components. The programs were mainly open source and free for research purposes. The connection of the software components was made with Java. Figure 13 illustrates the software tools used and their relations. The implementation was not suitable as such for use in industrial settings, at least not 24/7/365, but enabled us to perform experiments with real industrial data.

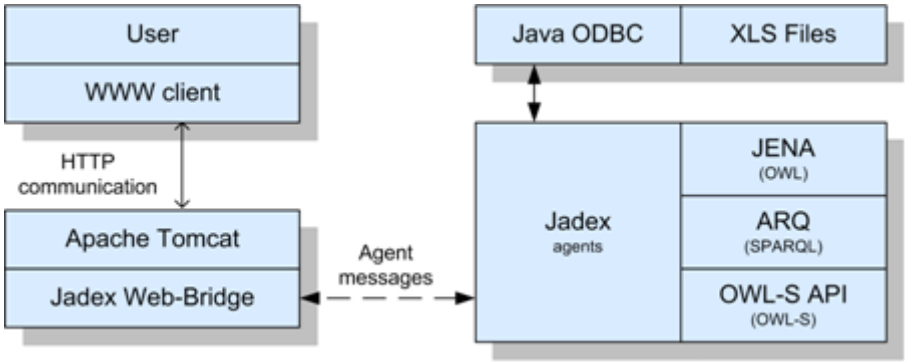

Figure 13 – Software tools used and demonstration system structure for experiments.

All the functionalities were built round the agent platform, which was selected to be Jadex in our experiments. Jadex is a general-purpose Java-based agent framework that naturally supports the use of the BDI model, and at the time of this research it seemed to be the most advanced BDI agent platform available as open source. The framework offers a relatively easy XML-based declarative configuration of agent functionality, and of course direct support for agent message sending and negotiations. More information about the Jadex agent framework can be found in Jadex (2008).

In addition to the agent framework, the system had multiple software modules with more focused purposes. The Jena (2008) Semantic Web framework was used to handle ontology-based world models, which were used as a basis for agent knowledge (especially in Experiment 2). Jena supports the novel ontology language OWL (OWL 2008) as an information format. The ARQ query engine for Jena was used for the searching and processing of ontology-based information (ARQ 2008). The ARQ engine supported the SPARQL language (SPARQL 2008) directly; this is similar to the SQL query language used with relational databases, and it was used in the experiments.

Furthermore, a number of additional software tools were used, but these did not have such an important role as those mentioned earlier. As the OWL-S standard (OWL-S 2008) was used for service description, the OWL-S API tool (OWL-S API 2008) was used in implementation. For the Experiment 2 web client interface, a Jadex Web-bridge Add-on (Jadex Web-bridge 2008) was used for the easy development of web-based access. Figure 13 also shows the Java ODBC bridge that was used to access the XLS files that contained the real data originating from industry that were used in the experiments.

6 Experiments illustrating the usefulness of agent design methods and developed architecture

6.1 Introduction

This chapter presents concrete examples of how flexible process monitoring functionalities may be designed with the Prometheus agent design methodology and using the specified agent architecture. The selection of the functionalities is based on the process automation user needs discussed in Chapters 2, 3, and 4. With these demonstration experiments the applicability of an agent-based design methodology and the architecture that was developed for constructing novel monitoring functionalities is evaluated and validated. The aim is to gain understanding about the feasibility of the methodology and architecture, not to specify exact blueprints about how the demonstrated functionalities should be realised. The experiments are related to the monitoring of measurement data, searching for process-related data, and the creation of symbolic data from measurements. The experiments are presented on a level of detail that shows relevant aspects related to the use of an agent-based design methodology and to provide a basis for discussion about architectural design.

In this thesis the experiments presented are:

- *Experiment 1: Temporal monitoring with constraints* – user-configurable monitoring of process automation measurements and their relations.
- *Experiment 2: Process event search* – searching for event-type data entries and combining these with related time series data.
- *Experiment 3: Change point occurrence monitoring* – monitoring change point activity on a user-defined group of measurements.
- *Experiment 4: Change point pattern monitoring* – observing user-specified change point patterns on-line from measurements.

With each experiment the design of the agents and the functionality itself is presented with a discussion of how well the methodology used suits this particular problem. With the first two experiments (1 and 2) a concrete implementation was made and the functionality of the experiments was tested with data gathered from real industrial processes. The latter two experiments (3 and 4) were designed and the discussion is based on this design phase, as there were no suitable data available for testing purposes.

Table 5 collates the descriptions, objectives, and properties of the experiments. As can be seen from the table, the four experiments cover the handling of information-processing tasks from different perspectives, and therefore provide a basis for a broader assessment of the agent-based approach. Furthermore, at the end of this chapter a concluding discussion is presented and the general applicability of the approach is summarised.

6.2 Design of experiments

With each experiment the agent design is presented. This provides a basis for the operational activity of the experiments and it also gives guidelines for partitioning functionalities into distinct elements. This part provides definitions and the responsibilities of agents, sequences of actions, and a description of the agent interaction protocols that were designed. The agent design is based on the Prometheus methodology, introduced in Chapter 3.2.4. However, the author has exploited the freedom of using only those parts of the methodology that are most suitable for designing the structural aspects of the monitoring system. With Prometheus these are the *System specification* phase, covering the initial development issues, and the *Architectural design* phase, used to define the general agent system architecture. The *Detailed design* phase was not used in the design of the experiments, because it does not cover structural issues and it is more closely related to implementation techniques. Furthermore, the developed agent architecture discussed in Chapter 4 and Chapter 5 is used as a basis for the design and implementation, and it provides a skeleton on the basis of which the operations of experiments are implemented.

In addition to the agent design part, every experiment provides discussion about issues related to decision-making. This part of the design varies between the experiments, depending on what functionalities the system aims to provide for the user. Furthermore, each of the experiments was developed, designed, and constructed for an individual and specific purpose and they have their motivation in industrial settings. In Table 5 below, the properties of each experiment are collected together. The mode support column refers to the work “modes” classification proposed by Paunonen (1997), discussed in Chapter 2.1. The column also shows if the functionality of the experiments supports on-line or off-line monitoring.

Table 5 - Collection of descriptions, objectives, and properties of the experiments.

Experiment name and description	Objectives and problems	Solution and properties	Mode* support and monitoring type
Experiment 1 Temporal monitoring User-defined timely restricted monitoring of process values representing good process state	Operators' interest is in the relative values of process quantities. Value changes may occur all around the system. Maintenance effort is minimised, as the functionality needs only a list of quantities and constraints related to them to be configured.	Distribute the monitoring, use agent negotiations to make monitoring agreements, monitor on the lowest possible level, and use the directory service to find monitoring service providers. Functionality does not need maintenance, and is possibly scalable?	The functionality is motivated by the performing predefined tasks (M2) mode, but is also usable for monitoring (M1) purposes. Usability in controlling disturbance (M3) mode is open. The experiment realises on-line monitoring.
Experiment 2 Process event search Search of related process events	User is interested in data that are available in separate data sources and in mixed syntaxes and formats. The viewpoint on the data stored in these sources also varies. Supervising user is interested in connections/links.	Use common base vocabulary and process model. Searching agent has description of data sources, and local data source adapts the data to a common format. Extendable, as user defines the search.	The functionality is motivated by the controlling disturbances (M3) mode and is usable when performing predefined tasks (M2). Usability for normal monitoring purposes (M1) is open. The experiment realises off-line monitoring.
*Experiment 3 Change point** occurrence monitoring* Process activity monitoring based on amount of changes visible in a user-defined group of measurements	User needs to be notified about amount of changes in a defined set of process measurements. Observe change point events from raw measurements. User specifies the list of monitored measurements, and system should monitor occurrence of change point events in this group.	Symbolic change point events are generated from measured numerical data on the lowest possible level, event notifications are subscripted, and monitoring manager monitors the occurrence. User defines the group of measurements and notification level; no maintenance is needed as operation is based on user configuration.	The functionality is motivated by normal monitoring (M1) mode. Usability in performing predefined tasks (M2) and controlling disturbances (M3) modes are open. The experiment realises on-line monitoring.
*Experiment 4 Change point** pattern monitoring* Process event monitoring based on user-defined activity pattern monitoring	User is interested in change point pattern monitoring, and the system should notify if the defined pattern has occurred in the system. Observe change point events on the basis of raw measurements. User specifies the monitored measurements and interest in event patterns related to these.	Symbolic change point events are generated from numerical measurement data on the lowest possible level, event notifications are subscribed, and manager monitors the pattern formation. User defines monitored patterns and measurement so no maintenance is needed.	The functionality is motivated by normal monitoring (M1) and is also usable when performing predefined tasks (M2). Usability for controlling disturbances (M3) mode is open. The experiment realises on-line monitoring.

*The modes are the following and the classification is based on Paunonen (1997):
M1: monitoring, M2: performing predefined tasks, and
M3: controlling disturbances (e.g. abnormal situations).

** The concept of Change Point (CP) is explained further in Chapter 6.5.3.

The subsequent chapters present the design for each experiment in the following overall and general structure. The ***Introduction*** represents the motivation and general background for the illustrated experiment. It also presents an example use case scenario describing how the functionality of the experiment could be used in a real process-related situation. The ***agent design*** shows relevant design artefacts based on the Prometheus methodology (discussed in Chapter 3.2.4). *System specification:* First, Prometheus is used to define the agent system goals, functionalities, and actions. Selections and naming in this phase are based on general background knowledge and the definition of the experiment. Then the methodology is used to outline the example given of a use case operation in a form of scenario, demonstrating the sequential relations between functionalities. In addition, percepts, actions and data is specified. *Architectural design:* In the next phase the generic agent roles, defined in the agent system architecture phase (Chapter 4.4), are used as guidelines for the clustering of functionalities to agents and to define agent descriptions. Finally, details of suitable interaction protocols for the functionality of the experiment are specified. The system overview diagram specified by Prometheus is not presented because issues related to definition of agent system boundaries were discussed in the architectural design (Chapters 4 and 5) and it does not present new aspects of agent design. Experiment 1 describes the use of Prometheus methodology more extensively, e.g. it shows how the methodology guides making of design decisions. The represented agent designs in experiments 2, 3 and 4 are more straightforward, concentrating on illustrating the final deliverables of each design phase.

The content of the ***functionality design*** part varies between experiments and it presents the methodologies used and tools that implement the decision-making. Then, for the first two experiments, the ***implementation*** part represents key issues related to implementation and shows the graphs and user interfaces that were designed for the experiment. The actual implementation of the system was based on the Jadex agent framework, as illustrated in Chapter 5.4. Then finally, at the end of every experiment the ***discussion*** summarises the results of the experiment, gives feedback on the architectural design, and also discusses the industrial potential of the functionality.

6.3 Experiment 1 - Temporal monitoring with constraints

6.3.1 Introduction

The main motivation for this experiment is a situation in which the process operator needs assurance that certain process values stay relatively to each other within specified limits. Typically, this kind of monitoring is needed temporally when performing predefined tasks, for example in the process start-up phase or a grade change situation. The temporal nature of these tasks is the reason for the name of this experiment. The design here focuses on the use case of a predefined task because in these situations the operator has much to do and monitoring help seems to be needed. Typically, the monitoring for these situations is relatively easy to configure, for example with describing the relations and limits for a couple of process quantities.

Example use case for the functionality:
A rule of thumb in a certain process industry site tells us that the operator should check that in a particular process area the current dose of chemical 1 is more than that of chemical 2. This simple check shows that the process is relatively OK, although there is no direct control loop connection between these two. Therefore, the operator wants to be able to set up a check (possibly more than one) that results in a notification if the specified relation is not valid.

In this experiment, it was decided to use formalism taken from constraint satisfaction problems (CSP) in the monitoring task definition. In short, CSP consists of a set of variables with restricted value domains and a set of specified constraints which the values of these variables must satisfy. Then value ranges satisfying the constraints are searched for with different types of search procedures. In this experiment, the operation is the opposite. Here, the constraints are used by the user to describe the normally-behaving process, as far as the selected and monitored variables are concerned. A user-centric approach in parameter setting was selected for this experiment, because a lot of knowledge about predefined tasks is said to be possessed just by the operators (Paunonen 1997). When monitoring, the system checks that the defined constraints are still valid every time a new measured value is available. Inference-making based on the numerical values of process quantities using CSP is discussed more thoroughly in Seilonen et al. (2006)

6.3.2 Agent design for temporal monitoring

The *system specification* design of the temporal monitoring experiment is started by defining the system goals. On the basis of the background idea and the described example use case it can be seen that the configuring, monitoring, and reporting functionalities are the functionalities needed in this experiment. First, the system is configured by the user with the defining constraints, and then the agent-based system performs the monitoring task autonomously and finally, when needed, the broken constraints are reported to the user. By naming and further refining these functionalities we get the following system goals.

System goals:

- CSP configuring – User defines monitoring functionality with CSP formalism, defining measurements, their values and relational constraints.
- CSP monitoring – Let peers always check constraint validity when new measurement data are available. Use the CSP engine to distribute the monitoring to peers on the lowest possible level where all the necessary data are available.
- Reporting – Report broken constraints to user.

Next, these system goals are broken down into functionalities. Figure 14 shows the actions and goals and how these relate to the functionalities in this experiment. In *CSP configuring* the goal is to *Define monitoring task with CSP* and the related action is *Activate CSP monitoring*. This functionality is triggered by the user. In the *CSP monitoring* functionality the triggering event is the new measurement value that has been detected and the related action occurs when one or more of the constraints have been broken. The *constraint consistency* is related to the checking of one constraint whose values are directly accessible, and *CSP consistency* refers to a set of individual constraints, where values are queried from numerous agents.

Furthermore, the *Reporting* functionality is responsible for notifying the user with a report about a broken constraint. The selection of actions, goals, and their names is based on the system goals presented above.

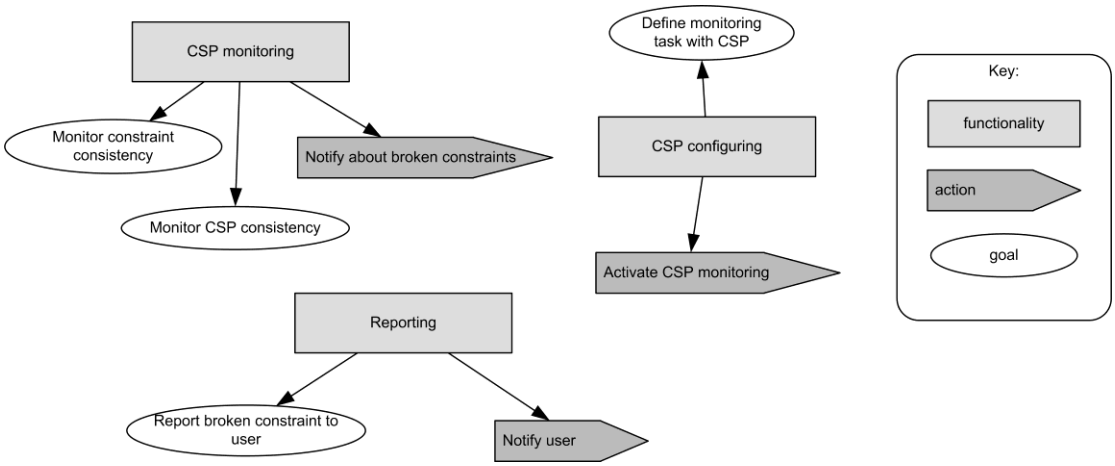

Figure 14 – Temporal monitoring functionalities with Prometheus notation.

The next step in the development is scenarios that describe how the goals and actions are linked together as a sequence. These sequences show the normal operation of the scenario, and they can reveal if an important part of the whole functionality is missing. Below is the *Temporal monitoring scenario*, which presents the normal operating sequence for this experiment, in which the user activates the monitoring task, and the set constraint validity is monitored until broken constraints are reported to the user. This sequence results in a functionality that is the one described in the use case of this experiment.

Outline specification of Temporal monitoring scenario:

1. GOAL: Define monitoring task with CSP
2. ACTION: Activate CSP monitoring
3. GOAL: Monitor constraint consistence
4. OTHER: Wait for measured values to change and break constraint
5. PERCEPT: new value in monitored measurement
6. OTHER: Check constraint validity
7. ACTION: Notify about broken constraints
8. GOAL: Monitor CSP consistency
9. OTHER: Check CSP validity
10. ACTION: Notify about broken constraints
11. GOAL: Report broken constraints to user
12. ACTION: Notify user

For OTHER type steps, see discussion in Chapters 6.3.3 and 6.3.4.

The next stage in the design of experiment is that the interaction of the system with the environment is covered. Perception in this experiment interfaces with the system in two sets of circumstances. First, when the user is defining the monitoring task and thus requiring the system to start monitoring, this should start the above-defined sequence. The user's deactivation of the monitoring should also be handled. These relate to the user activity percept, which is the triggering event for the whole experiment. The processing of user actions is the responsibility of the *user interaction* module, discussed in Chapter 5.2.5. Second, when processing the actual monitoring task, the system is responsible for always making a new inference when a *new value* is available in the monitored process measurements. In other words, the *new value* percept activates constraint consistency checking in the CSP monitoring functionality.

Internally, the system uses a number of actions to partition the operation, as may be seen in the scenario above. However, user notification is the only action that is seen from the outside, and the other actions can be seen as services requests used internally between the modules. The data produced by the functionalities, transferred between them, and also used internally by the functionalities is an important part of the system. In this experiment the following clusters of information can be identified:

- CSP – contains user-configured CSP constraints that define the monitoring task. The CSP formalism used is discussed in the next subsection.
- Measurements – contains process measurement data in time series format. These may be read from the underlying automation system with a special interface (e.g. OPC) or may be fetched from some dedicated process database.
- Broken CSP reports – contains records of broken CSP constraints. These reports are shown to the user with a suitable user interface.

After the system functionalities, sequences, and interfacing are defined the *architectural design* is started with grouping these elements to form agents. In Prometheus these groups are called agent types and they are a major part of architectural design. The grouping is done in iterative fashion, considering alternatives, and the final version of this design phase is shown in Figure 15. When selecting how to group functionalities to agents there are no right or wrong answers, but low cohesion and high coupling are objectives of clustering. Generally good guidelines are that agents' functionalities seem to relate, agent type "makes sense", and the defined agent can be described with a couple of sentences. Especially useful is to try to find a suitable name for an agent that intuitively holds the respective functionalities together. Also if two or more functionalities use same data extensively, then it is practical to group these together so that less communication between agents is required. (Padgham and Winikoff 2004). Generic agent roles presented in architectural design (Chapter 4.4) are used to guide the clustering.

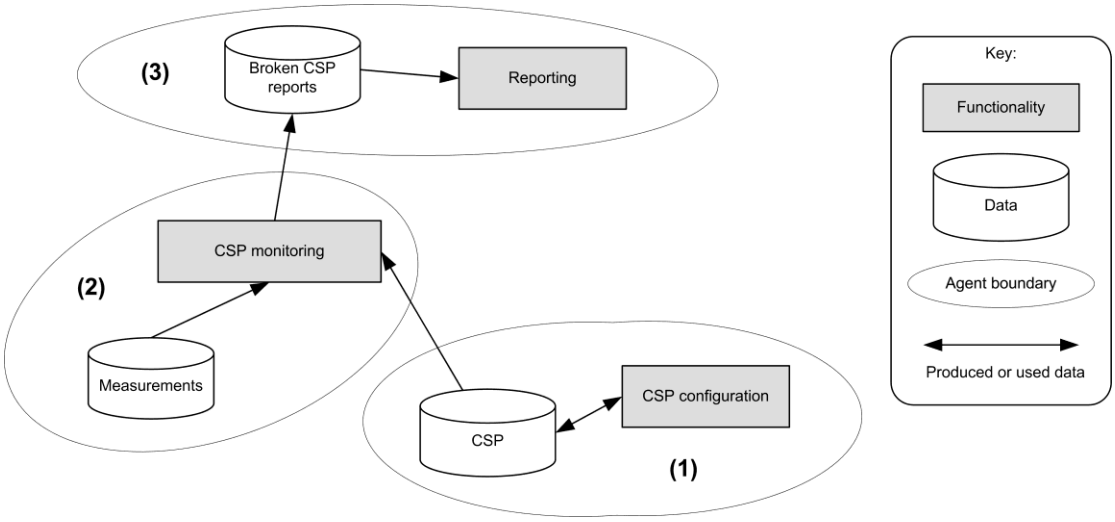

Figure 15 - Grouping of functionalities in temporal monitoring experiment using notation from Prometheus. The clusters are named the Configuring agent (1), Monitoring agent (2), and Reporting agent (3).

In temporal monitoring the clusters of functionalities are defined as follows:

- **Configuring agent:** The first cluster (1) holds the *CSP configuration* functionality used by the user to set up the monitoring task by defining it into the *CSP* configuration data store. Typically, there is one agent of this type per user. This agent is based on the C*lient agent* architectural role.
- **Monitoring agent:** The second cluster (2) holds the CSP *monitoring* functionality. This operation has to be active for the system to be able to observe changes in data. This functionality gets its initialisation from the *CSP* configuration data store. The number of these agents is not restricted, and it depends on the number of observed measurements and the hierarchical configuration of the monitored measurements within the physical process. The clustering of these agents is based on the *Information agent* and *Process agent* architectural role definitions.
- **Reporting agent:** The reporting cluster (3) involves the *Reporting* functionality, which is responsible for reporting the broken CSP constraints to the end user. Typically, there is one agent of this type per user. This agent realises the *Client Agent* role from the architectural role definitions.

When looking at the clustering of elements more closely, the question of merging Clusters 1 and 3 together may arise. These agents are kept separate in this design because their operational activity levels differ and they use different data. The configuring agent may provide a rather advanced user interface to help the user in the task definition phase, e.g. possibly detailed data models of the whole process are used. On the other hand, the reporting interface may be only a simple pop-up showing the broken constraints. Leaving the reporting agent separate and as simple as possible gives the opportunity to vary the reporting method, e.g. email and short message notifications could easily be used. Integration to the current control room alarming functionality may also be performed.

Specifying interactions: Interaction protocols capture the dynamic aspects of the system. First interaction diagrams are developed based on the described use case, outlined scenarios and specified agent descriptions, by replacing each functionality in the scenario with a responsible agent and further inserting communication between agents where it is needed. Then these diagrams are generalised to interaction protocols, e.g. as is shown in the following two figures below. In the temporal monitoring scenario the interaction protocol is divided into two natural parts, and Figure 16 illustrates the initialisation part, which is performed at the beginning of the performance of every monitoring task. The initialisation protocol shows how the initial values are first obtained from the responsible monitoring agents and then reports on broken constraints are requested from these agents.

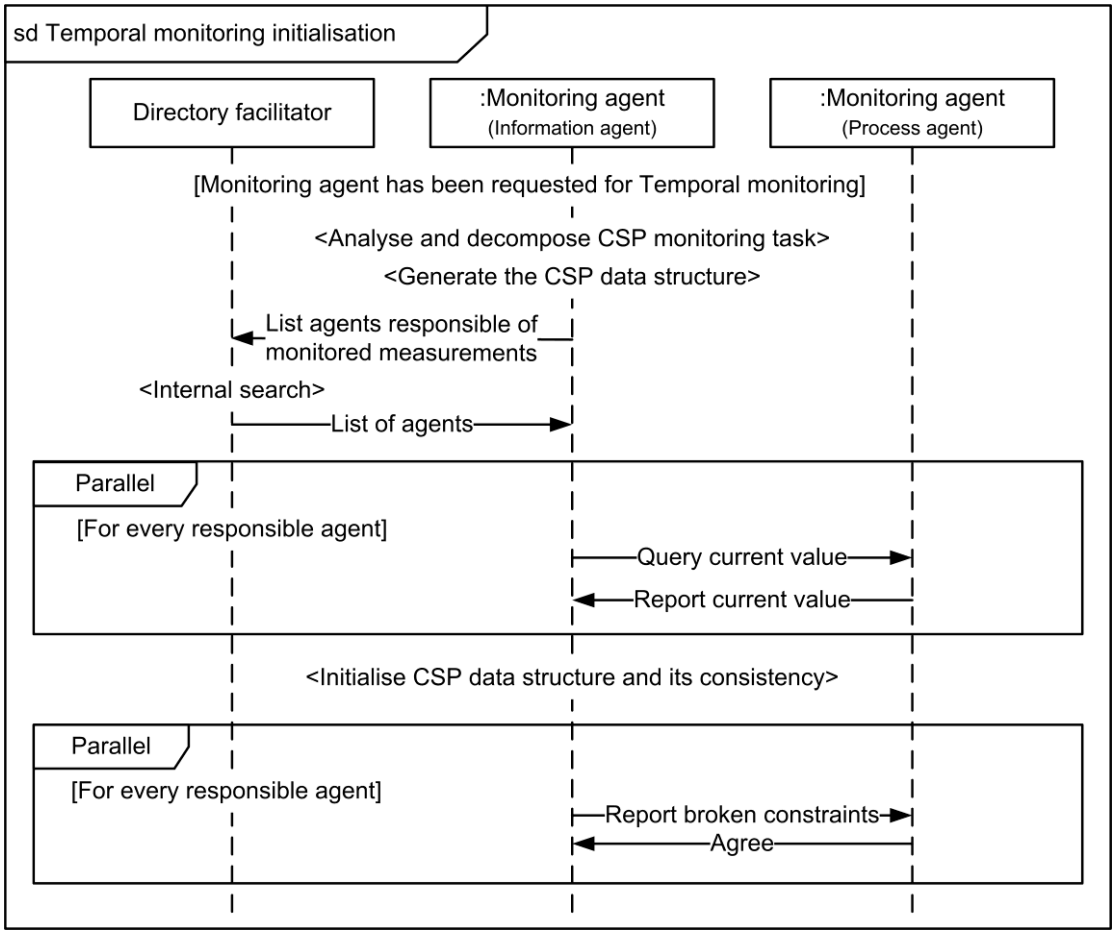

Figure 16 – Temporal monitoring initialisation interaction protocol using Prometheus notation.

Figure 17 shows the temporal monitoring interaction protocol. This illustrates the fact that the initialisation phase is performed once, directly after the user has requested the temporal monitoring of values. This is the part of the interaction that is responsible for the actual monitoring functionality. There the monitoring agent receives broken constraint reports every once in a while and then checks if the constraint situation is still satisfactory. If not, then the user is reported. This operation is repeated until the user deactivates the whole CSP monitoring task.

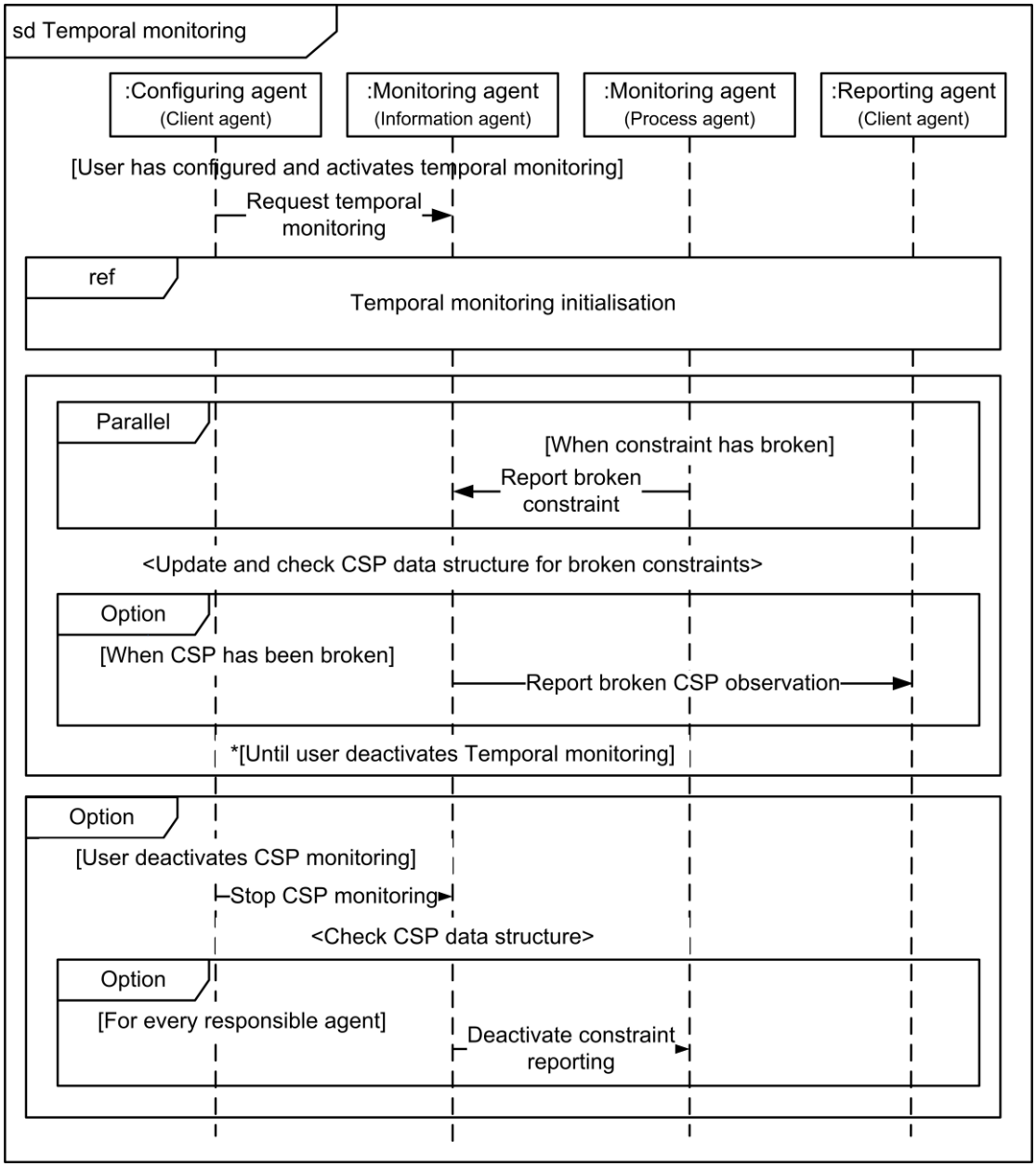

Figure 17 – Temporal monitoring interaction protocol using Prometheus notation.

The presented interaction protocols illustrate how agent communication is used to produce the normal operation of the temporal monitoring experiment defined by the example use case.

The illustrated agent design demonstrates the use of the Prometheus methodology to define the structural skeleton for the temporal monitoring system. In temporal monitoring the role of agent design is strong because the monitoring activity is mostly handled by the agent system. The actual decision-making, based on checking the consistency of constraints, is clearly visible in the interaction protocols and also in the clustered agents. The benefit of using agent interactions is that it naturally delegates and distributes the monitoring operation and thus contributes to the adaptation to dynamically changing situations.

6.3.3 Constraints for process condition monitoring

In this experiment constraints were used as a method to model user-originated monitoring logic and express it to monitoring agents. The user describes the desired behaviour of the process with constraints, and the fulfilment of the defined constraints is monitored by the agent system, as was shown in the previous chapter. The principle of using constraints in process monitoring described in this chapter was presented originally, and in more generic form, by Seilonen et al. (2006).

The introduction for this experiment stated that the user is interested in finding out if the current process situation and respective values are in conflict with the user's knowledge about what those values should be when everything in the process is OK. To be able to use constraint satisfaction problem (CSP) formalism for monitoring purposes, suitable constraint templates describing the relationships between measurements needed to be selected. In this experiment the constraint templates used were Binary and Unary constraints, and the whole monitoring task was configured by defining a set of these constraints with a logical and-operator between them.

The Binary constraint is in the form <*VAR1 (operator) VAR2* (comparison) *value*>. In our model for binary constraint the *operator* may be one of the following ('+', '-', '*', and '/'), and the *comparison* is either greater or less than. Here the *VARs* refer to some specific physical measurements and the *value* is some defined arbitrary numerical value. The Unary constraint template is simpler and it is a form of <*VAR* (comparison) *value*>. Table 6 and Table 7 show both constraint templates with practical examples. The constraint model is known to be limited and extending it is a possible part of future development.

Table 6 – Binary constraint template and respective examples.

	Constraint	**Example usage**
Template	VAR (operator) VAR (comparison) VALUE	
Example1	Pump1Speed + Pump2speed > 10	Two pumps are outputting to the same tank and the total input value should be at least something
Example2	Consistency * Pump2Speed > 120	Mass flow should be over the limit
Example3	Pump1Speed / Pump2Speed > 1	The speed of Pump1 should be more than the speed of Pump2

The Binary constraint gives the user a flexible way of defining various process-related constraints. The Unary constraint is practically identical to the alarm functionality within present process automation systems, and does not as such offer any additional functionalities to the user. However, offering both types of constraints gives the user flexibility in defining the whole monitoring task.

Table 7 – Unary constraint template and an example.

	Constraint	**Example usage**
Template	VAR (comparison) VALUE	
Example1	Temperature < 70	The temperature should be less than 70

The CSP data structure referred to in the agent interaction protocols (Figure 16 and Figure 17) contains a set of user-defined Binary and Unary constraint templates explained above.

6.3.4 Implementation and tests

The motivation for the Temporal monitoring functionality had its background in an industrial need. The test scenario concerns the pH control in the bleaching of mechanical pulp in a paper mill. This is one of the key process variables operating in hostile conditions – exposure to vibrations, corrosive chemicals, and wide temperature variations – which makes the measurements vulnerable and subject to malfunctions. Therefore, the additional monitoring was to be developed to aid operators in being notified about potential problems in pH measurements. To demonstrate the functionality of the experiments to non-researchers an implementation of the system was developed in the PROAGE project. The software tools used and the structure of the system that was used in this experiment were presented in Chapter 5.4. The demonstration itself was constructed as a standalone system that worked with offline data. The data were samples from a real production plant imported to the system by means of XLS files.

The user interface (see Figure 18) was designed to be an add-on to current control room displays, and not to be a standalone and whole new application. The interaction of the tool was based on the idea that the operator could drag and drop items from the picture to the monitoring user interface. This setup was simulated with screenshots taken from a real production environment and by realising operational hotspots to these pictures the user could drag the items to be monitored by the tool. The monitoring tool window was started from a separate pop-up menu. After the variables had been dragged to their place in the monitoring configuration window, the operator had to define the operator, comparison, and compared values to each constraint template.

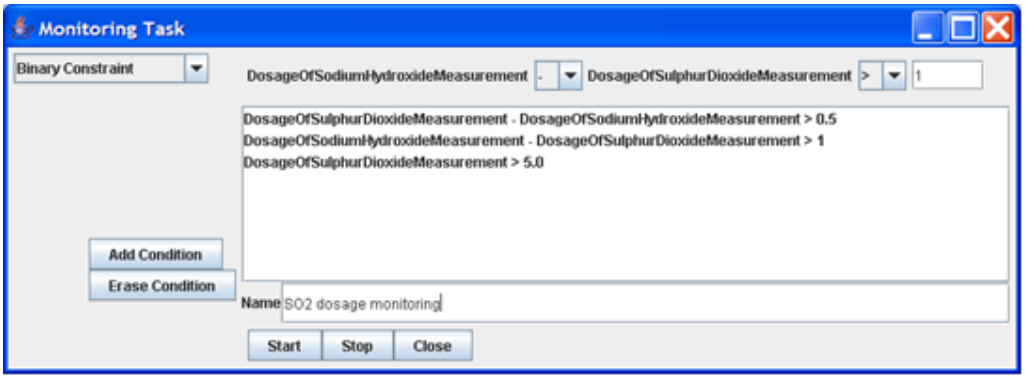

Figure 18 – Screenshots from the demonstration user interface for setting up the temporal monitoring task with binary and unary constraints.

After finalising the monitoring setup definition, the user names the task and sends it. In the demonstration system there was no direct method to see what was happening when the monitoring was running, but the debugging tools offered by the Jadex framework could be used if needed. With these tools the user was able to see what messages agents were sending to each other and it also gave the opportunity to trace how the constraint variables values changed over time. The demonstration user interface for reporting was just a message box showing the broken constraints with the values that broke them.

6.3.5 Discussion

Using the Prometheus design formalism and specified agent architecture presented above seems to support the construction of temporal monitoring functionalities with CSP formalism. Distributing constraint checking to multiple agents using agent negotiations was straightforward, as agent communication standards provide suitable protocols for distribution. Agent negotiation provides a natural way to construct an operation that adapts and responds to dynamically changing process situations and the respective value changes. However, the Prometheus does not address the issue of how to select a suitable FIPA standardised interaction protocol for the task, as it focuses on describing suitable communication from functional perspectives. This more abstract level of defining communication is appropriate for the application domain specialist that does not have an agent technology background. However, within the conducted implementation and tests the *fipa-subscribe* (FIPA 2002) was found to be well suited.

Basically the idea behind this experiment is to generate a user configurable and extended version of alarm functionality provided by current process control systems. Based on the work done in the experiment it can be stated that many process-related monitoring tasks can be defined using CSP formalism. Using CSP as a basis for monitoring makes the execution engine extendable, but CSP formalism is not something that a process operator naturally understands and can utilise directly. Although UI aspects were not studied in this research, it may be stated that a more user-friendly input methodology should be provided if CSP is to be used in industrial settings. The real feasibility of this kind of end user focused functionality is strongly influenced by usability issues, so further study should focus on e.g. studying user activity in a larger number of use cases within real industrial settings.

The temporal monitoring functionality was primarily designed for situations in which predefined tasks are performed and the supervising users need tool support for gaining information about significant changes in the process. However, the temporal monitoring tool could also be used in a normal monitoring situation as well, for example as a more flexible way to configure alarm functions or, in scheduled runs, to check that the relations between the most important process values are as wanted. Temporal monitoring may also be used when dealing with disturbances, e.g. noticing a disturbance from known symptoms and preventing the disturbance from occurring again. The continuous development may also benefit when an alert on any interesting special process situations may be obtained and possibly complex value combinations may be monitored.

This experiment was the first that was designed and implemented so it dealt mostly with agent negotiations and interaction protocols. However, the functionality of this experiment would probably be rather easy to add to current systems as the user provides all the definitions for decision-making and thus the monitoring would not need additional configuration work in the form of process models and deduction rules. Although the presented approach offers extended functionality compared to currently available alarm solutions, the real applicability and benefits of the solution presented here depend on the industrial settings. As measurements are typically stored to one centralised database there is no reason to distribute the monitoring either. However, if in the future field devices, distributed IO equipment, and process stations provide for added functionalities to be implemented in those, then the agent approach presented here seems to be reasonable.

6.4 Experiment 2 - Search of process events

6.4.1 Introduction

This experiment has its motivation in the situation where an operator is accessing stored history data and trying to find interesting bits and pieces of information. Typically, this happens when the user is trying to control disturbances and searching for possible symptoms from the stored data. Within sites important process-related information is stored in separate databases with mixed presentation formats, and most of the time any connection between past process events and measurement time series is not easily evident. What is also challenging is that there is an inconsistency in viewpoints about how a certain process element is shown, how it is stored, and what data are linked to each element. For example, for the maintenance personnel a process device is a physical object with mechanical properties, but for the operator that same device is important from the functional perspective.

Example use case for the functionality:
Quite often a user wants to find out what has happened in the past in a certain process area, e.g. what happened in the previous 8-hour shift in the area that the operator is responsible for. Typically, the operator starts the shift by going through information that is stored in a variety of data sources to find out out-of-the-ordinary events that might influence the way process operates. As most external data sources use mixed naming practices, varying data presentation formats, and separate user interfaces, the user needs to go through them one by one. A unifying system that operates more automatically and gathers important events together and finds relations between events would help the user to generate a general view of the situation.

Differences in viewpoint influence the user interface and this, for example, makes searching for interesting relations between process events laborious. For the supervising user the interest is typically in some process area that is decomposed spatially or functionally and the elements that are related to it. Therefore, one purpose of this experiment was to produce a system that would provide the user with one summarising interface to search for and view “all” the information that is related to a specified process element. The basic idea is to use the speed and accuracy provided by IT to go through information stored in separate data stores and delegate the combination work to the system. Technically, this experiment uses Semantic Web tools to link elements in various data sources and a more thorough discussion about these aspects can be found in Pakonen et al. (2007). The following section focuses on illustrating how agent design methodology can be used to develop this kind of functionality.

6.4.2 Agent design for search functionality

The design of search of process events starts by defining the system goals on the basis of the introduction and example use case. For this experiment the initial system goals are search definition, searching, and result viewing. First, the search is activated by the user by defining the search parameters. Then the system searches for the specified information from various data sources and combines the result, which is finally shown to the user. By naming and further refining these we get the following system goals:

System Goals:

- Search definition – User defines search parameters with a suitable interface that has some knowledge about the underlying process setup and capabilities of the available data sources.
- Search decomposition – As data are stored in various data sources, the search needs to be decomposed into parts that the respective data sources are able to answer.
- Searching – Matching sub-results are searched for in individual data sources.
- Result combining – Combine overall result on the basis of partial sub-results using formal data models that describe how an element in one data source sub-result relates to elements described in other sub-results.
- Result viewing – Search results are presented to user.

Next, these system goals are broken down into functionalities and activities. Figure 19 shows how the actions and goals are related to the respective functionalities in this experiment. The functionalities that are visible to the user are *Search definition*, responsible for activating the whole search operation, and *Result viewing*, responsible for showing the results to the user. The actual search operation consists of the *Search decomposition*, *Searching*, and *Result combination* functionalities, which organise the search operation. *Search decomposition* is divided into a part that is responsible for searching through currently active and available data sources from the system and a part that decomposes the search into partial search definitions. The *Searching* functionality searches for data from data sources and the respective action delivers the sub-results. The *Result combination* combines sub-results and sends them to be shown to the user.

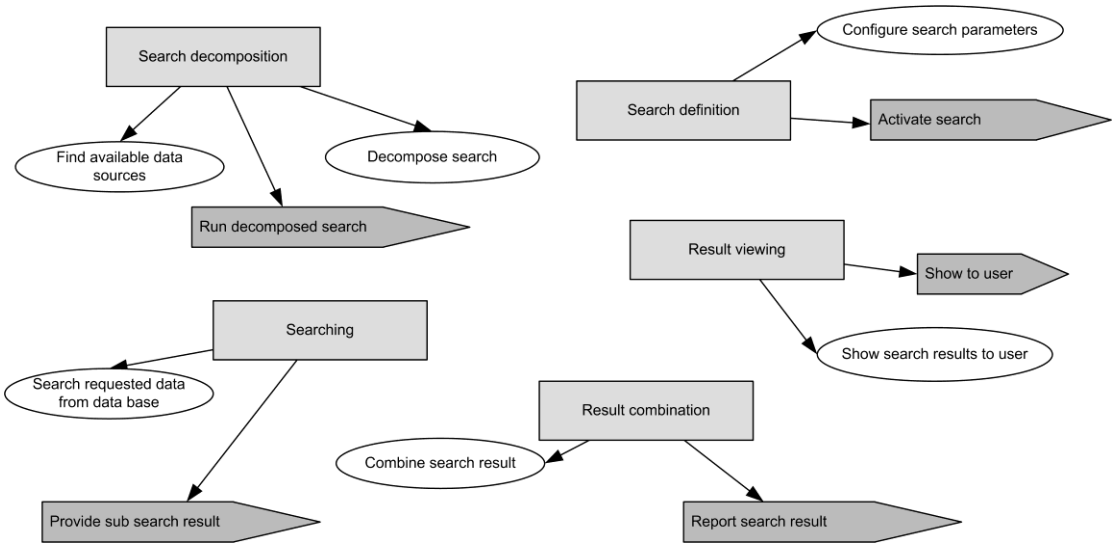

Figure 19 – Search of process event functionalities using notation from Prometheus.

The next progress step is scenario development, which describes how the goals and actions are linked together in the form of a sequence. Below is the *Search for process events scenario*, which presents the normal operating sequence for this experiment in which the user defines the search, the system finds relevant data from various sources using a network of information-providing elements, and then the system merges the partial results into a combined final search result. The presented scenario is activated by the user. This scenario illustrates system operation that responds to the example use case described in the introduction to the experiment.

Outline specification of Search of process events scenario:

1. GOAL: Configure search parameters
2. ACTION: Activate search
3. GOAL: Find available data sources
4. OTHER: Use service descriptions to find responsible agents
5. GOAL: Decompose search
6. OTHER: Use models to partition the query
7. ACTION: Run decomposed searches
8. GOAL: Search for requested data from data source
9. OTHER: Format sub results to common format
10. ACTION: Provide sub-search results
11. GOAL: Combine search results
12. OTHER: Use data models to link individual events
13. ACTION: Report search results
14. GOAL: Show search results to user
15. ACTION: Show to user

For steps with OTHER type specified see discussion in Chapters 6.4.3 and 6.4.4. Next in the design of experiment, the interface descriptions are presented. As this experiment is activated by the user and then the system performs the search autonomously, there are no other percepts than observing the first user activity that triggers the whole operation. Internally, the system uses a number of actions to partition the operation, but showing the results to the user is the only one that is visible outside the system. Furthermore, in this experiment the following data stores are used to exchange information between agents and their functionalities:

- Search – contains user-defined search. The search and its parameters refer to available process ontologies.
- Process model – contains instance types of information about process elements and their relations in the particular process. The properties of the process model are discussed in the next section (Chapter 6.4.3). Basically, the process model should be general to the whole system and available to all players.
- Sub-result – contains records of data that are the results of queries to individual data sources.
- Search result – contains the overall search result that is shown to the user.

Specifying agent types: After the system functionalities, operational sequences, and environment interaction are defined it is possible to group these elements into meaningful agents, using the generic agent role definitions discussed in Chapter 4.4 as guidelines. Figure 20 illustrates the final version of this iterative grouping process.

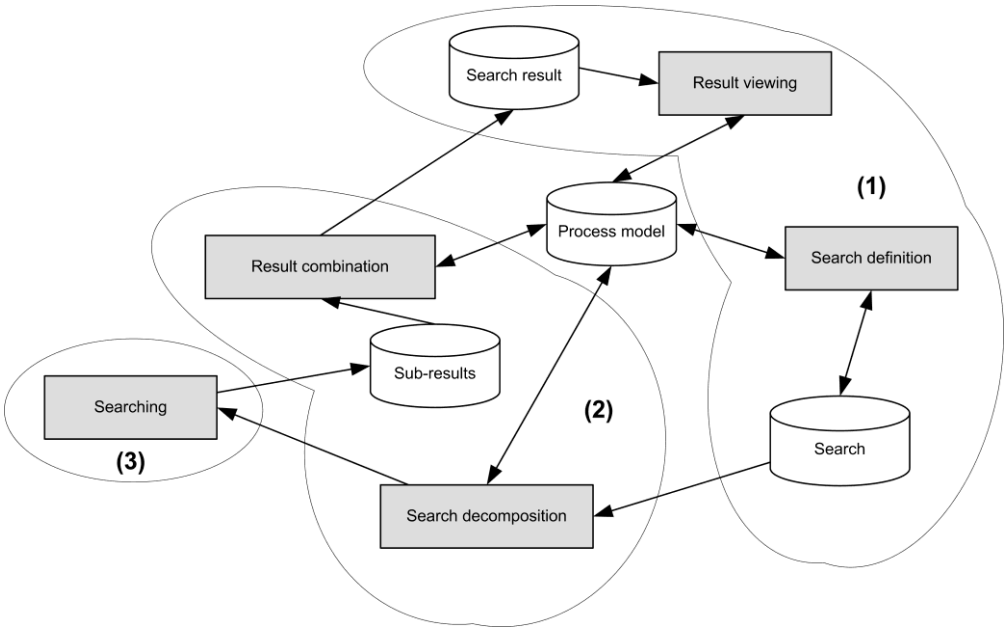

Figure 20 – Grouping of functionalities in search for process events experiment with Prometheus notation. The clusters are named Client agent (1), Search agent (2), and Database agent (3).

In Search of process events the clusters of functionalities are defined as follows:

- **Client agent:** The first cluster (1) groups the user interaction, which is partitioned into the *Search definition* and *Result viewing* functionalities. From these functionalities the *Search definition* is used by the user to produce *Search* data. *Result viewing* is responsible for presenting the *Search result* to the user. *Process model* is used to provide search options to the user and support the formatting of the results. These functionalities are event-based, and they are activated when the user defines the search and search results become available. In general, both these events are rather rare, and typically one agent per user is enough. This clustering uses the *Client agent* architectural role definition.
- **Search agent:** The second cluster (2) involves search managing functionalities, namely the *Search decomposition* and *Result combination* functionalities. Search decomposition uses *Process model* and *Search* data and delegates searching to responsible parties. *Result combination* uses *Process model* data when constructing an overall search result. The number of this type of agents is not defined in advance, and it may change dynamically, depending on the search requests. This clustering is based on the *Information agent* and *Wrapper agent* architectural roles.
- **Database agent:** The searching cluster (3) builds the searching functionality and handles one data source. This functionality answers queries that are related to a certain, possibly legacy, data source. The number of these agents depends on available data sources, as typically there should be one agent responsible for each data source. The architectural role of the *Wrapper agent* is used to guideline this agent.

In this experiment the user interface, acting as the input and output channel between the user and the system, is defined as being handled by one agent. This is mainly because it is best to realise the handling of complex data models once and in one place. The process model data are left outside the individual agents in the clustering because in the design of this experiment they are classified as global knowledge that is available and common to every participant operating in the system.

The search functionality was partitioned into two different agent types. Wrappers (Database agents) were designed to handle the formatting of inconsistent data to a common and understandable format. This was seen to be a distinct operation from the actual search decomposition and fusing functionality, which was left to the Search agent.

Specifying interactions: As the operation of a search experiment on the agent interaction level is rather simple, the Interaction protocol for process event searching is easy to construct. Figure 21 shows the Search of process events interaction protocol that was developed.

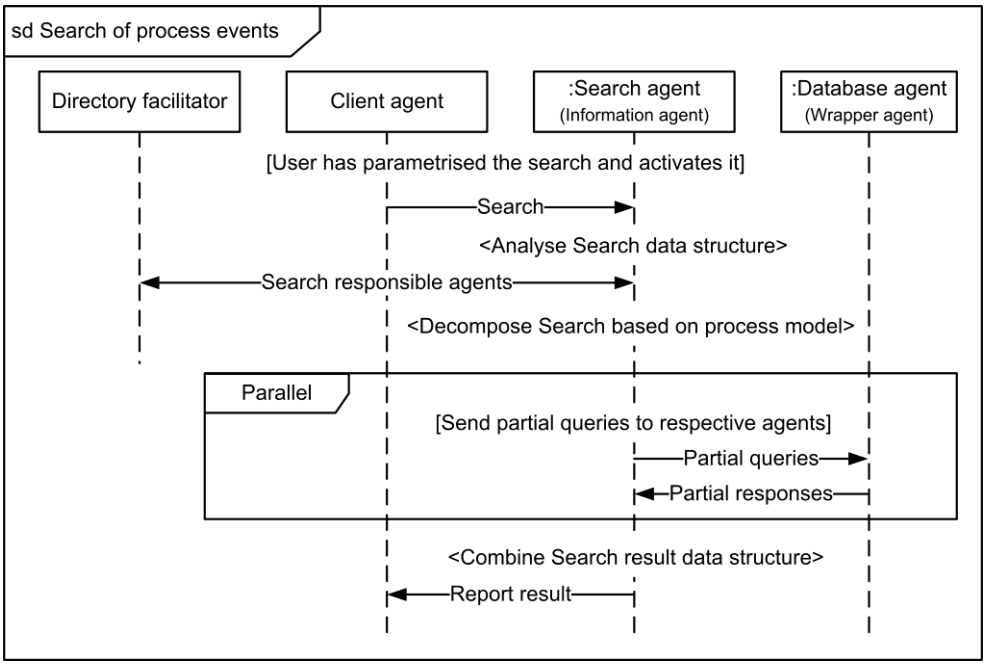

Figure 21 - Search of process events interaction protocol using Prometheus notation.

The illustrated interaction protocol realises the main use case scenario of the search functionality described in the introduction to the experiment.

The agent design defines the structural skeleton for the operation behind this experiment. In Search of process events the benefits of agent design are more visible in system structuring than in defining operations with agent negotiations. This is partly because the search activity is heavily based on straightforward procedural operation, and the focus is on utilising Semantic Web tools to handle data modelling issues. Nevertheless, the use of systematic agent design assists in the partitioning of the operation.

6.4.3 Data models for information retrieval

To be able to query and fetch data from multiple data sources requires every agent in the system to communicate with each other using a common, understandable, and agreed vocabulary. In a software system this vocabulary should also be machine-processable and agreed at least between the agents which are exchanging information among one another. As was discussed in Chapter 3.4, this may be realised with software tools originating from the Semantic Web. In this experiment we decided to use a solution in which every entity in the system is responsible for providing information using a common vocabulary defined in a shared ontology, and furthermore data model introduced domain division was used in service coverage definitions.

The concepts in the common ontology, called a plant ontology in this experiment, were selected to describe important process-related matters from the operators' viewpoint. The spatial relations of objects were found to be important, especially because process operators are typically responsible for some physically restricted part of the process. In this experiment the plant ontology is used in both cases, when searching for the stored data and also when the user is describing the query. In the query definition phase, the ontology is used to guideline the construction of the query in such a way that it is understandable and processable by the agent who receives it. The query decomposition was based on domain definitions, as each wrapper was responsible of a certain domain of knowledge. In the search phase, the plant ontology describes how things are related to each other, so that the search-processing agent may find relations from the partial answers that the *Wrapper agents* sent to it. Figure 22 shows an example of how concepts from different viewpoints can be connected when physical domain knowledge is used.

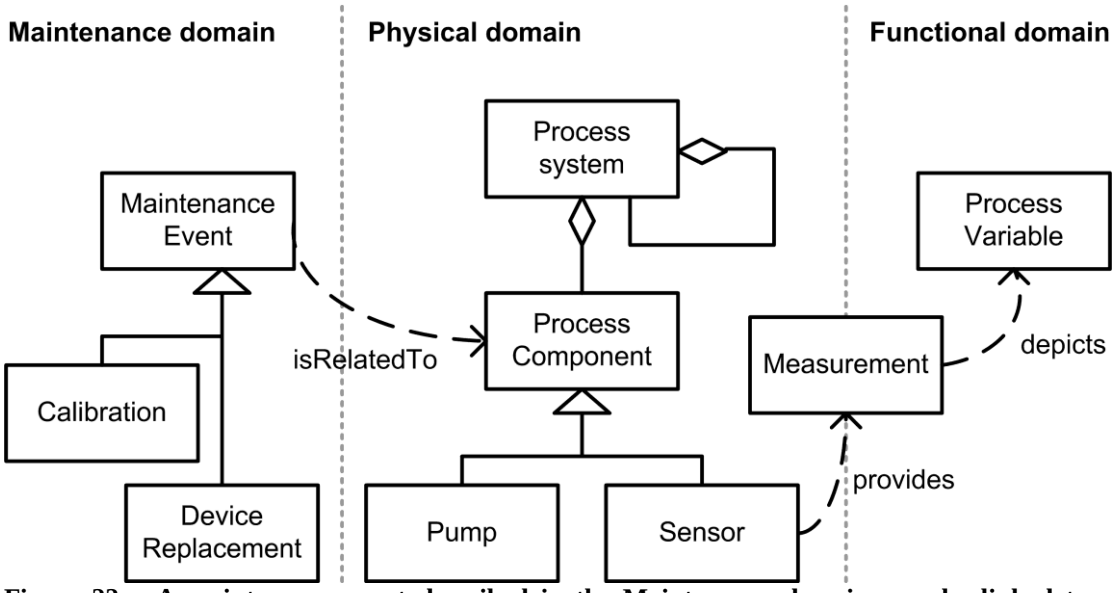

Figure 22 – A maintenance event described in the Maintenance domain may be linked to a certain Process variable described in the Functional domain when using object relations defined in the Physical domain ontology. (Pakonen et al. 2007).

In this experiment, the spatial relations of process automation concepts were utilised. It was decided that every *Wrapper Agent* would be responsible for providing extra information that described how their objects link to objects in the Physical domain. This idea seemed to have a rather intuitive background as things in process monitoring tend to happen somewhere in the process. For example, as may be seen above (Figure 22), if one agent provides information on how a maintenance event is connected to a process component and another expresses the connection of a process variable to a sensor, then the relation between the maintenance event and a certain process variable may be found. More thorough discussion about ontological aspects of the experiment 2 and architecture can be found in Jussila (2006) and Pakonen et al. (2007).

6.4.4 Implementation and tests

A working demonstrator for this experiment was developed in the PROAGE project and implementation was performed by the project team. A browser-based user interface was chosen as the method to communicate with the user and the look and feel were designed to resemble those of an ordinary web search engine as much as possible. The structure that was realised, connecting the web server and the agents, was shown in Chapter 5.4. As a result of the practical limitations of the research project, our setup of experiments used off-line data dumps obtained from a real industrial process. The dumps used contained information from the plant measurement history, maintenance database, laboratory database, and an electronic diary.

The experiment use case starts when the user defines the search. Typically, the operator was thought to be interested in what events had occurred in the previous shift or in a longer time period. Another thing that the users were interested in was similar situations to the current one, trying to figure out how possibly to cope with the problem. Figure 23 shows the search user interface realised for demonstration purposes. It enabled the user to define the time span, process area, and event type that (s)he was interested in. The choices in the search parameterisation related to plant ontology and instance data that describe the particular process under consideration.

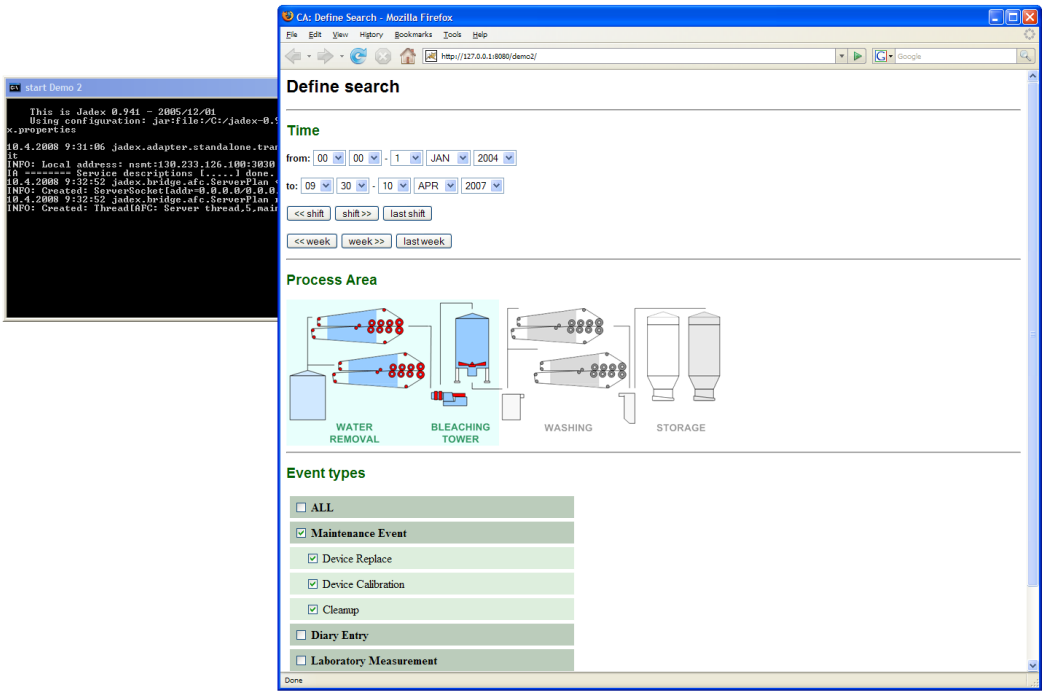

Figure 23 – User interface search definition in search of process events. Also visible in the "Define search" window is a process layout overview from the bleaching of mechanical pulp, which was the test environment for the experiment.

After the user specified the search and activated it the web server generated and sent an agent a message containing the search. In the implementation the search was expressed with the SPARQL language, which is similar to the SQL language used to query data from relational data bases (SPARQL 2008), but it is designed to query data from ontological data stores. OWL-S (2008) was used by Wrapper agents for service description and as each data source was holding data from a certain domain the service description was selected to be defined base on these. In concrete software realisation OWL-S API (2008) was used as an engine to match sub searches and responsible agents.

Then, after the agents have done their searches and the result has been combined, the outcome is shown to the user. Examples of these results are shown in Figure 24. The idea of the interface is to combine information retrieved from multiple sources in one combined user interface. In the result view the measured time series data were shown with the related event information. In this experiment these were maintenance events and diary entries. In addition, the laboratory measurements used for calibration purposes were also shown, when available, beside the time series data.

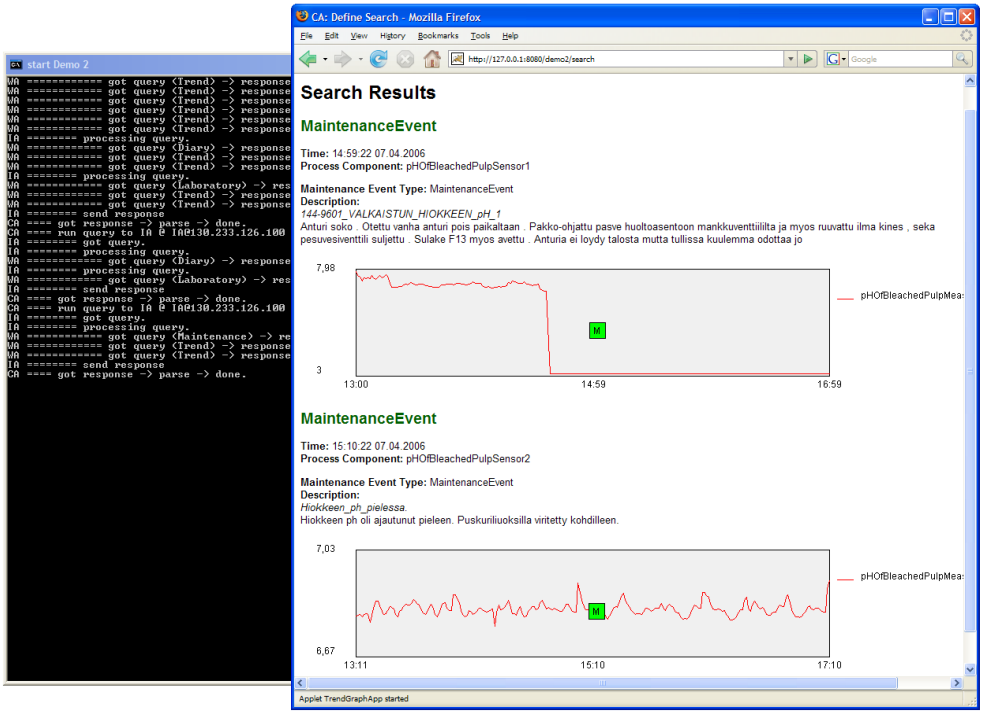

Figure 24 - User interface for results in search of process events.

The implementation was in many aspects very simplified and built only for demonstration purposes. In the future, potential targets for development would assist the user in the search definition phase and facilitate flexible navigation through the results. Iteration between search definition and result viewing should also be made possible.

6.4.5 Discussion

In this experiment the functionalities were based mainly on operations on semantic plant models and ontology engineering. The agent design was straightforward and provided structural guidelines for modular system design, but the agent design did not solve the knowledge-related issues. This issue is visible in an outlined scenario where numerous OTHER type steps are defined. The search as a functionality is procedural and does not require the autonomy or flexible communications that agent technology provides. The experiment proved the potential of Semantic Web tools when integrating multiple data sources and searching for links between elements in those sources. Ontological aspects of data processing within this experiment are discussed more thoroughly in Jussila (2006) and Pakonen et al. (2007).

This experiment provided users with a search tool for finding relations in process events and measurements. In general, the benefit of this type of functionality would be easier data access, which makes the examination of miscellaneous process history data reasonable even when specific problems are not being searched for. The main design focus in this experiment was to provide a tool to help operators to control disturbances. Nevertheless, searching for interesting phenomena from history data could also be a useful feature in normal monitoring mode and when performing preplanned tasks. With the developed tool a person could “surf” history data and find interesting events and possibly find the relations that these may have. And if interesting relations are found, these could be used as the basis for building new monitoring functionalities.

6.5 Experiment 3 - Change point occurrence monitoring

6.5.1 Introduction

Typical industrial processes are designed in such a way that when they are operating normally the physical process quantities are relatively steady and the trend lines visible in the control room are practically straight. On the contrary, when the values of measurements and actuators are changing more than normally then this might be a result of a problem. Paunonen was thinking similarly in his research (1997: 75-76) when pondering the possibilities of detecting process "fever" by "change sum" calculation. In general, several value changes occurring simultaneously in multiple quantities may be related to an activity burst that may be the result of changes in input material or equipment malfunction. Sometimes these bursts of changes are the result of desired set point changes that have been made elsewhere in the production site, e.g. when performing grade change.

Example use case for this experiment:
The process operator has learned that normally the values of a set of process variables are stable. Therefore, the operator has defined a list of 20 process quantities, for example measurements and actuators that are situated in some physical area, and requested the system to provide notification if more than 10 significant changes occur in a 30-minute time span.

The focus in this experiment is on the normal on-line monitoring of industrial processes. This experiment describes the development of a monitoring system that the operator can use to delegate the process "fever" monitoring. In this experiment this "fever" monitoring is based on the idea of monitoring Change Point (CP) events generated from measurement time series values. CPs are defined as time points at which the characteristics of process quantity change significantly, and these can be found from time series data with statistical tools (see the discussion in Chapter 3.5). Furthermore, the system is used to observe occurrence peaks in the amount of these events in a user-specified time window.

6.5.2 Agent design for change point occurrence monitoring

The design of change point occurrence monitoring starts with defining the system goals on the basis of the described use case. For this experiment these functionalities may be seen as configuring, change point observing, occurrence monitoring, and reporting. First, the system is started with the user configuring a group of measurements and setting alarm occurrence limits for the group. Then the responsible agent performs change point detection for individual process measurements. Then the change point occurrence for a specified group of measurements is calculated and, when needed, the exceeded limits are reported to the user. By naming and further refining these functionalities we get the system goals as follows.

System Goals:

- User configurability – User defines and groups a set of measurements to be change point monitored and specifies the occurrence peak limit for this group.
- CP observing – Observes change point in individual process measurement.
- CP occurrence monitoring – Observe occurrence peaks in group of measurements.
- User reporting – Report change point occurrence peaks to user.

Next, these system goals are broken down into functionalities, activities, and their relations, as shown in Figure 25. In this experiment the functionalities *User configurability* and *User reporting* are the ones that are visible to the end user. The actual monitoring functionality is split into two separate parts. *CP observing* is responsible for detecting change points from measurement time series and notifying observations. The actual observation is discussed in Chapter 6.5.3. *CP occurrence monitoring* gathers together the individual CP observations and monitors their timely amount within the user-defined group.

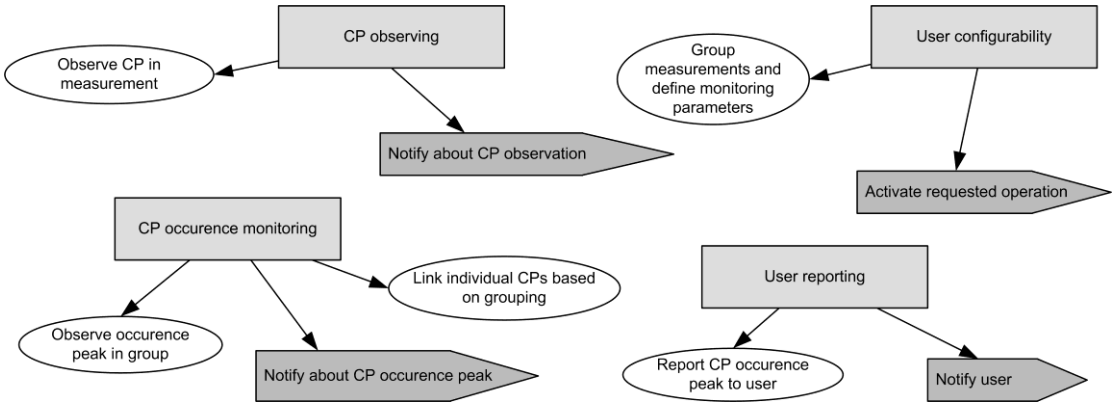

Figure 25 – Change point occurrence monitoring functionalities using notation from Prometheus.

The next step is the scenario development, which describes how the goals and actions are linked together as a sequence. Below is the *Change point occurrence monitoring scenario*, which presents step by step the normal operating sequence in which the user activates the occurrence monitoring task, the system observes change points and monitors the amount of them, and finally, when needed, the findings are reported to the user. This scenario produces the functionality described in the example use case of this experiment.

Outline specification of Change point occurrence monitoring scenario:

1. GOAL: Group measurements and define monitoring parameters
2. ACTION: Activate requested operation
3. GOAL: Observe CP in measurement
4. OTHER: Wait for process quantity to change substantially
5. PERCEPT: *CP observed*
6. ACTION: Notify about CP observation
7. GOAL: Observe CP occurrence peak in a group
8. OTHER: Check CP occurrence peak
9. ACTION: Notify about CP occurrence peak
10. GOAL: Report CP occurrence peak to user
11. ACTION: Notify user

See discussion in chapter 6.5.3 for the steps having the type OTHER defined.

The interaction of the system with the environment is the following. Perception is responsible for triggering activities in the case of *user activity* and when a *CP observed* is available in measurements. Internally, the system uses actions to trigger functions, and the only one of these visible outside the system is *Notify user*. Furthermore, the data used in this experiment are defined to be the following.

- *CP configuration and grouping* – contains information on what measurements are to be observed for CPs and how these are grouped together for occurrence monitoring. Also defines the time and amount of alarm limits.
- *CPs* – contains online records of observed process change points.
- *CP occurrence monitoring data* – stores history of observed CPs that belong to a user-defined group and are within a specified time window.
- *CP occurrence peak reports* – contains information about occurrence findings that exceed the defined alarm limits.

After the system functionalities, operational sequences, and environment interaction are defined it is possible to group these elements into meaningful entities (agents). The final version of this iterative grouping process is illustrated below (Figure 26). The clustering is based on the generic architectural role definitions presented in Chapter 4.4.

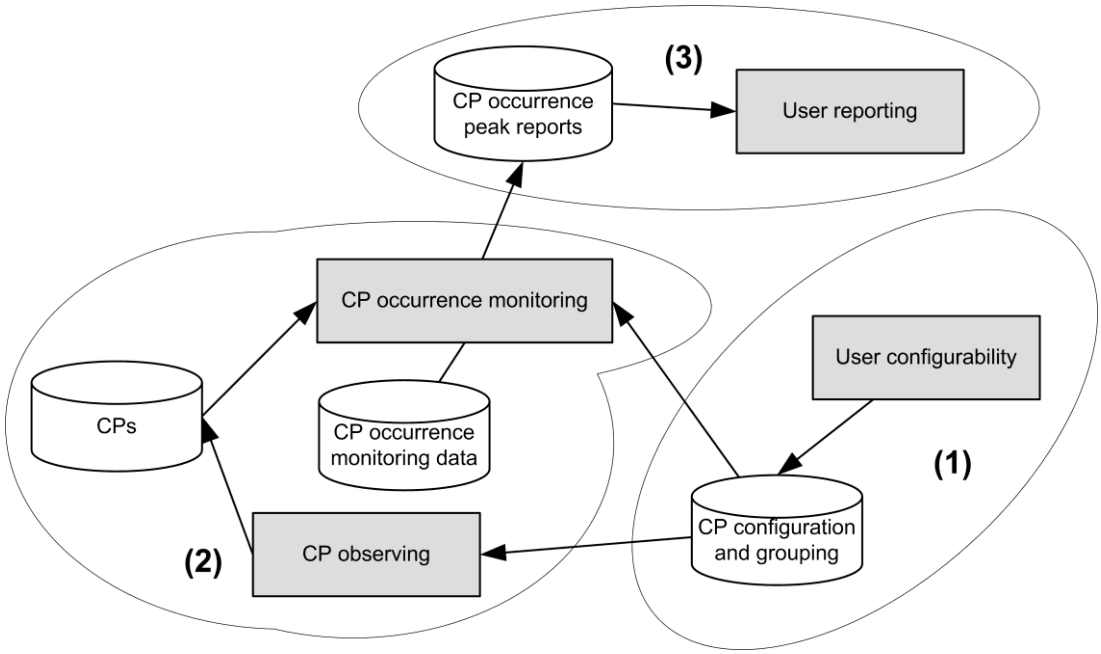

Figure 26 - Grouping of functionalities in change point occurrence monitoring experiment with Prometheus notation. Clusters are named Configuring agent (1), Monitoring agent (2), and Reporting agent (3).

In Change point occurrence monitoring the clustering is defined as follows:

- **Configuring agent:** The first cluster (1) holds the *User configurability* functionality, and produces *CP configuration and grouping* data. Typically there is one agent of this type per user in the system. The architectural role specified as the *Client agent* is behind this clustering.
- **Monitoring agent:** The second cluster (2) involves monitoring functionalities, namely the *CP observing* and *CP occurrence monitoring* functionalities. These are operations that have to be active in such a way that if something happens in the data, these functions should observe it. These functionalities get their initialisation from the *CP configuration and grouping* configuration. The number of these agents is not restricted, and it depends on the related process configuration and possibly on the number of observed measurements. This clustering is influenced by the architectural role definitions of the *Information* and *Process agents.*
- **Reporting agent:** The reporting cluster (3) involves *User reporting*, which is responsible for reporting the issues found in the system to the end user. This part of the system functionalities is event-based; they are activated when new user reports are stored in the *CP occurrence peak reports* data store. Typically, there is one agent per user of this type. This clustering realises the functionalities defined by the architectural role of the *Client agent*.

In agent definitions, a question about merging Clusters 1 and 3 together may arise. These two clusters are kept separate in this design because their operational activity differs and they use different data. The configuring agent may provide a rather advanced user interface, based e.g. on process functional and physical models, and it may also be integrated with other applications in the control room. On the contrary, the reporting interface may be only a simple display giving notification about occurrence peaks. Leaving the reporting agent separate and as simple as possible gives the opportunity to integrate it directly to e.g. process control system alarm functions.

Interaction protocol development is based on use cases, scenarios, and related agent division. Change point occurrence monitoring is divided into two natural parts. Figure 27 shows the initialisation performed at the beginning of every configured monitoring task. The initialisation protocol shows how the peers are first queried from the directory and then the peers are requested to report about change points that occurred in individual measurements. The initialisation also generates and initialises the CP occurrence monitoring data structure.

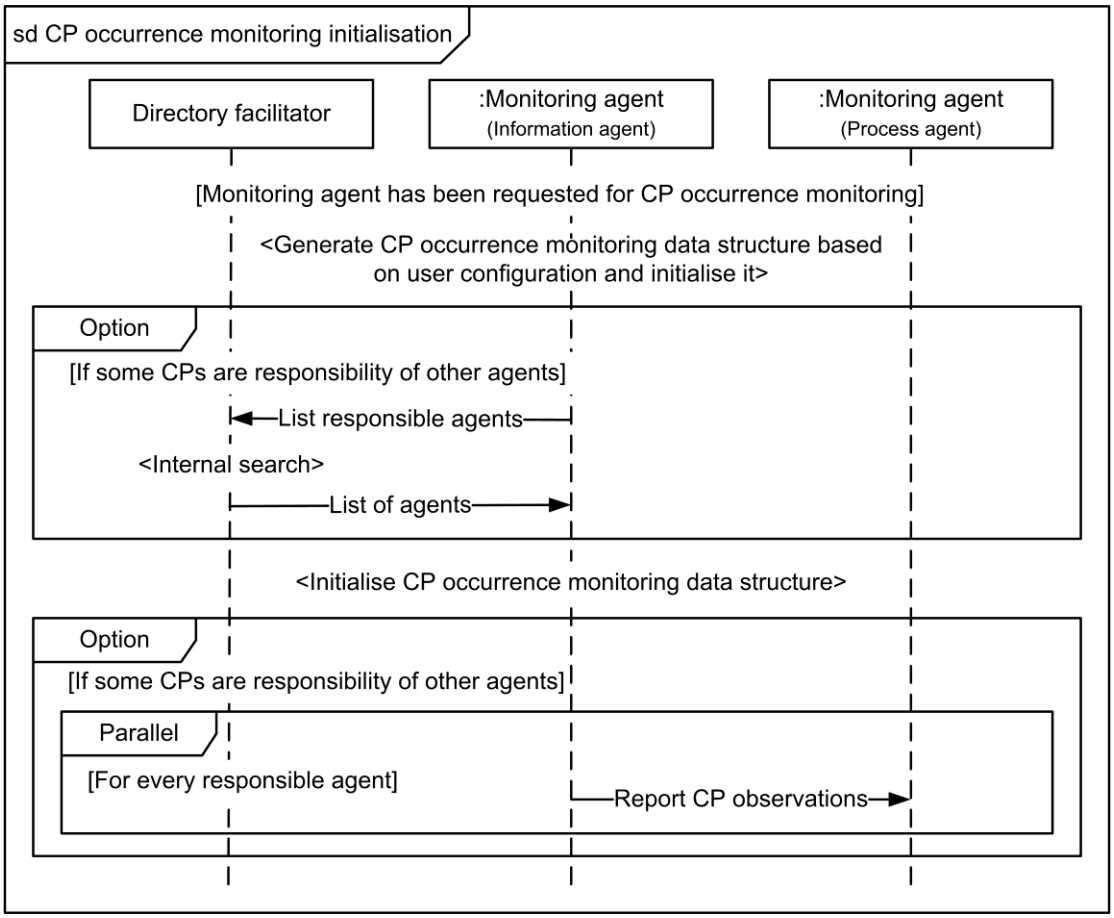

Figure 27 – Change point occurrence monitoring initialisation interaction protocol using Prometheus notation.

The change point occurrence monitoring interaction protocol is illustrated below (Figure 28). This shows that the initialisation phase is performed once, directly after the user has requested occurrence monitoring, and then the actual monitoring is performed. There the responsible agent receives notification about CP occurrences in the measurements that are of interest, and if enough change points are detected in the specified group and given time span then this is reported to the user. This monitoring operation continues until the user deactivates it.

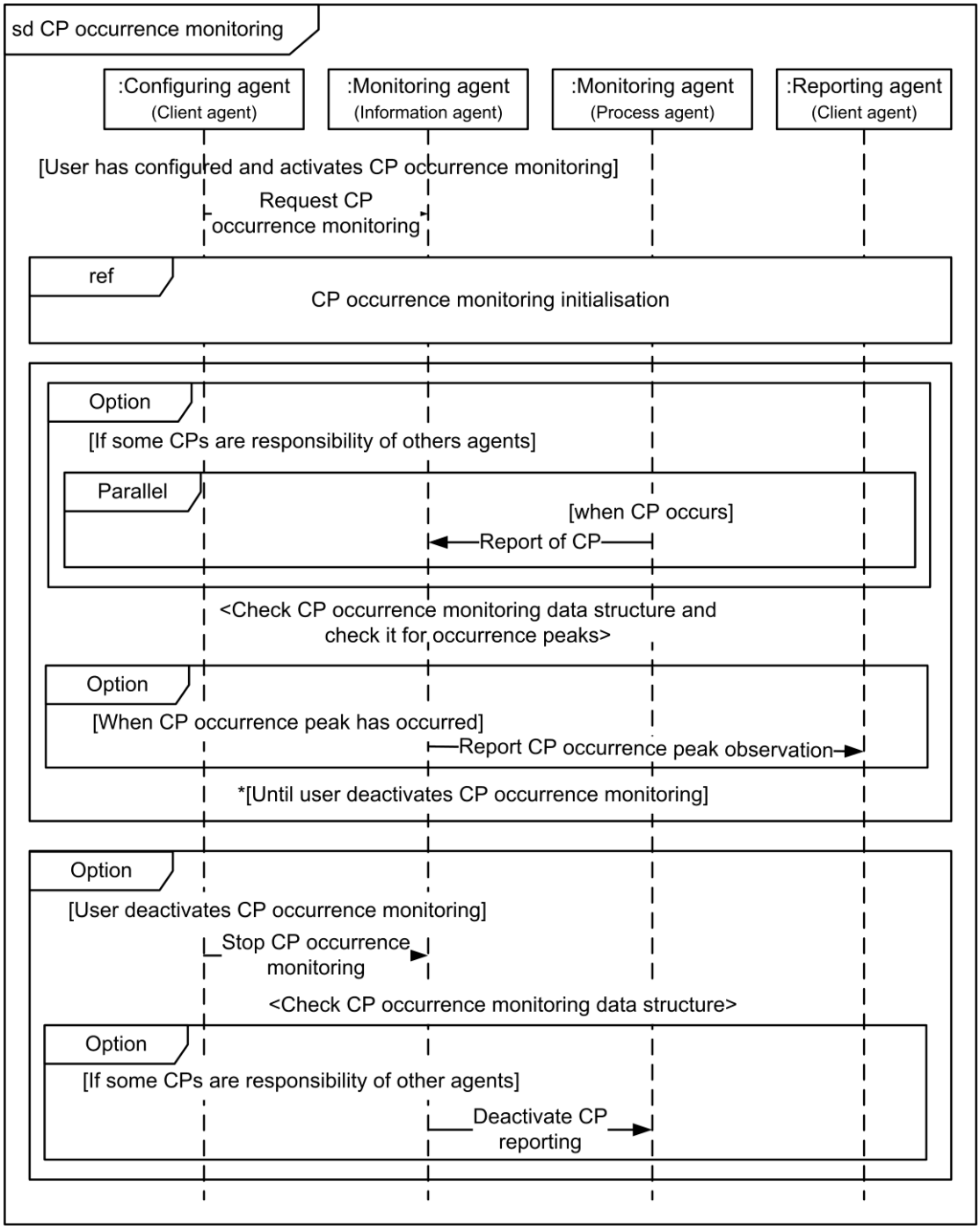

Figure 28 – Change point occurrence monitoring interaction protocol using Prometheus notation.

These interaction protocols together realise the main use case scenario presented in the introduction. In this experiment the role of agent negotiations is strong as they are used to organise and delegate the monitoring functionality.

6.5.3 Change point detection and occurrence monitoring

In this thesis a Change Point (CP) is defined as a time point at which the time series values of individual measurements change significantly. There are two distinct motivations for using CP detection. On the one hand, successful CP detection from noisy measurement data produces intuitively meaningful events and raises the abstraction level for an easier decision-making process. On the other hand, agent-

based systems operate most naturally with symbolic data and CPs offer a potential solution for converting process automation-related numerical data into symbolic form. In this thesis the target is not to design and implement the best possible CP detection algorithm but to find a suitable system structure that could utilise this kind of information for monitoring purposes. Therefore, a demonstrative realisation has been made in order to understand the applicability and limitations of CP detection.

In this thesis the actual realisation of the CP observation is based on the simple idea of approximating periods of individual time series values with a straight line. Figure 29 shows an example excerpt in which CPs are observed from time series values originating from real processes. The assumptions for the algorithm design are that the time series data are piece-wise continuous and the data contain white Gaussian noise. These data contain segments that are separated by points at which values change substantially, called CPs, and the segments are not necessarily related. In other words this means that the data may be seen as a series of linear segments, each having stochastic length and these segments may be approximated with a straight line.

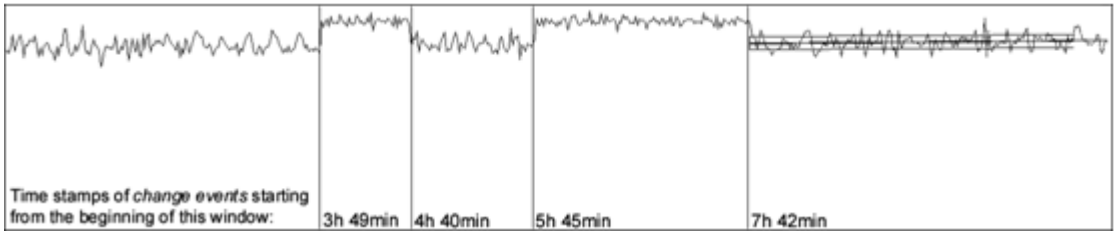

Figure 29 – Example excerpt of Change Point of observation process produced by demonstration implementation. The vertical lines show the points where the CP observation found significant changes in the measured values.

In our system the demonstrative implementation of this symbolisation function is called *Liner*. *Liner* fits lines to data by selecting the best straight lines that can be drawn backwards from the current time to the previous CP using least squares calculation as the evaluation criterion. The algorithm does not have any assumption about the variance of the values. Another important design requirement for *Liner* is that it needs to react to CP occurrences as they happen, on the so-called online principle. This is an important issue as the algorithm design cannot make an assumption that all the data are already available for processing e.g. in a history database, which is typically necessary for e.g. efficient *divide and conquer* clustering algorithms to be able to work. Finally, *Liner* is also thought to possibly operate on the device level, so it needed to be relatively efficient.

As defined in the use case, the user is thought to be interested in the amount of CPs in a certain physical process area and in a certain time window. Figure 30 shows an example in which it may be seen that the amount of activity in process measurements varies in time. The amount of variation depends on the selected measurements, the time window that is under consideration, and the activity of the process at the given time.

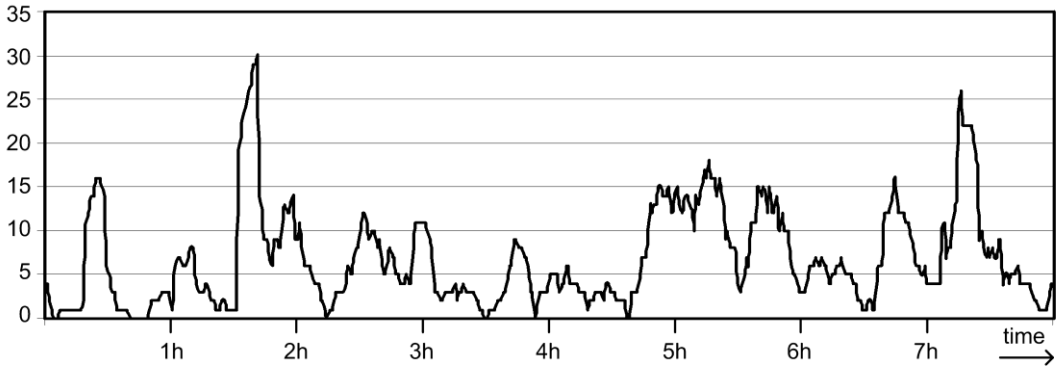

Figure 30 - Example graph of 10-minute moving average of amount of detected CPs in one-minute resolution from selected measurements. The amount of variables in the whole data set was 149 and altogether the graph represents an 8-hour time period. The selection of CP observed measurements was made on the basis of the process operational task and spatial relations. The test data were measured from the bleaching of mechanical pulp in a paper plant. The graph does not attempt to be a usable end result as such but it shows that the process activity level varies in time, as was discussed in the introduction to this experiment.

As a configuration for this experiment, the user needs to set up the occurrence monitoring by grouping the measurements and defining the monitoring parameters. Table 8 shows the configuration data structure and its attributes which were used in this experiment.

Table 8 - CP configuration and grouping data structure.

Parameter name	Type	Example values
Group identification	String	Group A
Monitoring time window	Integer, minutes	30 min
Notification level	Integer, events	8 events
Observed measurements	List of names	AB10045, AC10010, AD10011

When the actual monitoring is activated, it uses the following data structure (Table 9) to store CP observations. This data structure is used to check if the notification condition is achieved.

Table 9 - CP occurrence monitoring data structure.

Parameter name	Type	Example values
Group identification	String	Group A
Detected CP list	Ordered list of (time stamp, name)	17:14, AB10045 17:26, AB10045 17:42, AC10010 18:03, AD10011

The checking of the CP occurrence monitoring data structure is based on the following principle (Figure 31):

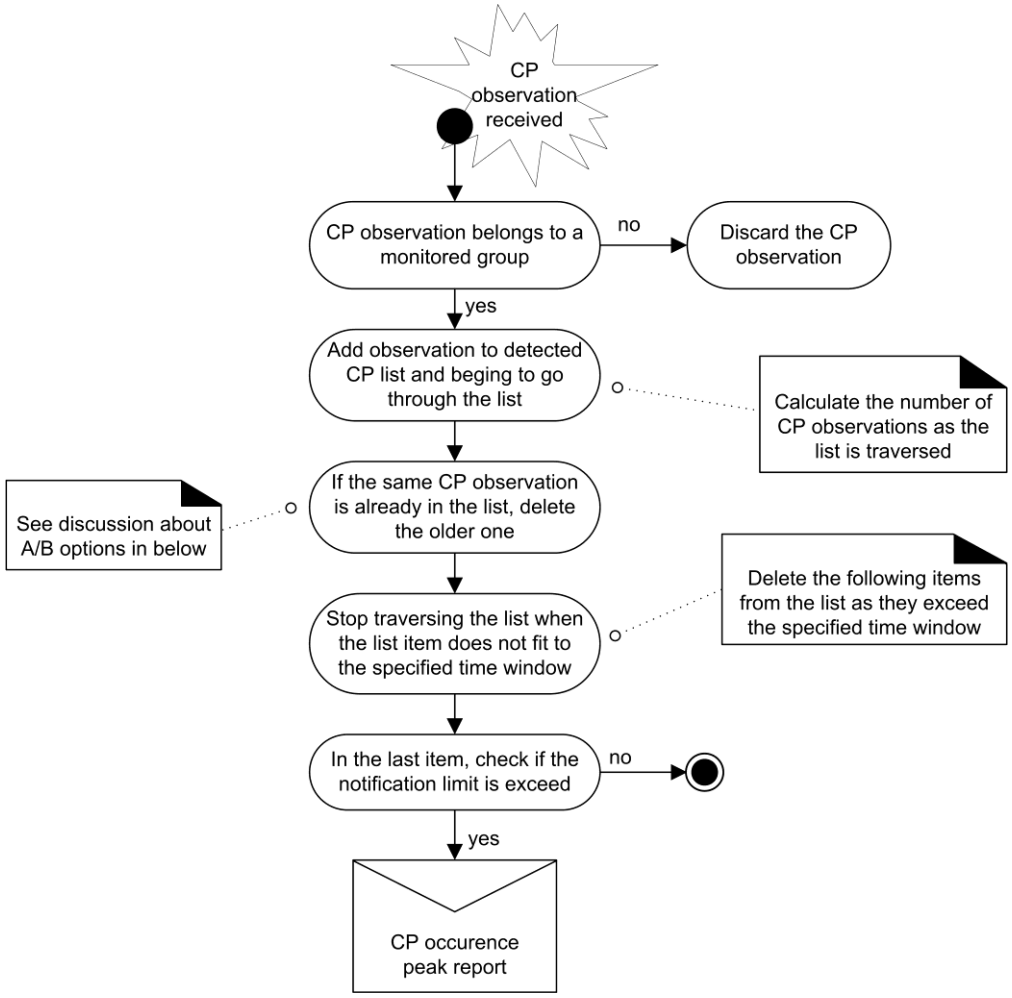

Figure 31 - Specification of CP occurrence monitoring activities, which shows in principle how an agent processes received CP observation. Notation is based on Prometheus.

When going through the list of detected CPs there are the following two options:

A. Count how many distinct elements have generated CP observation in the given time span. In this case, only one CP event is memorised for individual measurement, and rapidly alternating measurements cannot activate the notification. This is thought to be the default option in this experiment.
B. Count the total number of CP observations in this group within the user-specified time span. In this option, every received event is calculated and one individual measurement may produce more than one event.

The occurrence peak reports are a combination of *CP configuration and grouping* data with the copy of *CP occurrence monitoring data* that activated the reporting.

On the basis of the demonstration implementation of CP detection, it is possible to conclude that it is possible to construct this kind of detection and that it can provide results from industrial data with minimal configuration; see Figure 30, for example. This is important, as setting up monitoring agents within an industrial context should require as little configuration as possible for the overall experiment functionality to be feasible. In addition, the amount of CP points seems to vary with respect to time, so the functionality of the experiment is potentially usable in an industrial context.

6.5.4 Discussion

It is possible to develop the functionality of the experiment using an agent design methodology. The functionality is well suited to an agent-oriented approach, as agent negotiations provide tools to distribute the monitoring. Furthermore, the design illustrates how CP detection may be moved to the device level in the future if the devices permit this. On the basis of the short discussion about realising CP detection with statistical methods, it seems possible to state that it is possible to construct this kind of functionality and future research on the issues is proposed.

Change point occurrence monitoring introduces a new functionality that is not currently available in commercial process automation systems. The functionality is intended for monitoring mode within the normal operation of processes, but it is thought that it may also be useful when preplanned tasks are performed in production. The real value and usability of this kind of operation is unknown at the time of this research as the experiment was not motivated by industrial partners. For this functionality to prove its real potential it needs to be tested in an industrial setting.

6.6 Experiment 4 - Change point pattern monitoring

6.6.1 Introduction

Device malfunctions and other frequently-occurring problems may advance quite similarly from the first symptom to the realisation that there is a problem. For typical and regular problems these sequences may also be known and with corrective operations performed in time the resulting losses and damage to equipment may possibly be prevented. These types of problems have been studied extensively in the past; e.g. Fault Detection and Identification (FDI) research introduced numerous methods for practical usage (Chiang et al. 2001). The previous work has been noted in this research. In this experiment the basic idea is to use CP detection to build a user-configurable fault detection functionality.

Example use case for the functionality:
The user has found out that a certain process activity pattern typically leads to problems with process equipment or control. Therefore, the user defines a pattern of change points and requests the system to provide notification when this pattern has been observed in the process measurements.

This kind of monitoring operation would be helpful for the user in a normal monitoring situation, which is defined as monitoring mode by Paunonen (1997). In a normal and steady situation the process values do not fluctuate much and the operator may sometimes become bored. Generally, a steady situation also makes the change point pattern detection more reliable as there is no mixing noise present.

6.6.2 Agent design for change point pattern monitoring

The design of change point pattern monitoring starts with defining the system goals on the basis of the introduction and described use case. For this experiment these functionalities are configuring, CP observing, pattern monitoring, and reporting. First, the system is activated and configured by the user defining the interest of a certain change point pattern. Then the agents perform change point detection for individual process measurements. Then patterns of change point notifications are monitored and finally, when the occurrence of some configured pattern is observed, the user is notified about this. By naming and further refining these we get the following system goals.

System goals:

- *User configurability* – User defines measurements to be CP monitored, and sets up the patterns that should be monitored by the system.
- *CP observing* – Observe CP in individual process measurements
- *CP pattern monitoring* – Gather individual CP observations and observe CP pattern formation based on the user-configured patterns.
- *User reporting* – Report CP pattern findings to user.

Next, these system goals are broken down into related functionalities and activities, shown in Figure 32. In this experiment the functionalities *User configurability* and *User reporting* are the ones that are visible to the end user. The actual monitoring functionality consists of two separate parts, as follows. *CP observing* is responsible for detecting change points in the measurement value time series (see also Section 6.5.3). Then *CP pattern monitoring* gathers change point detections and recognising patterns from individual events.

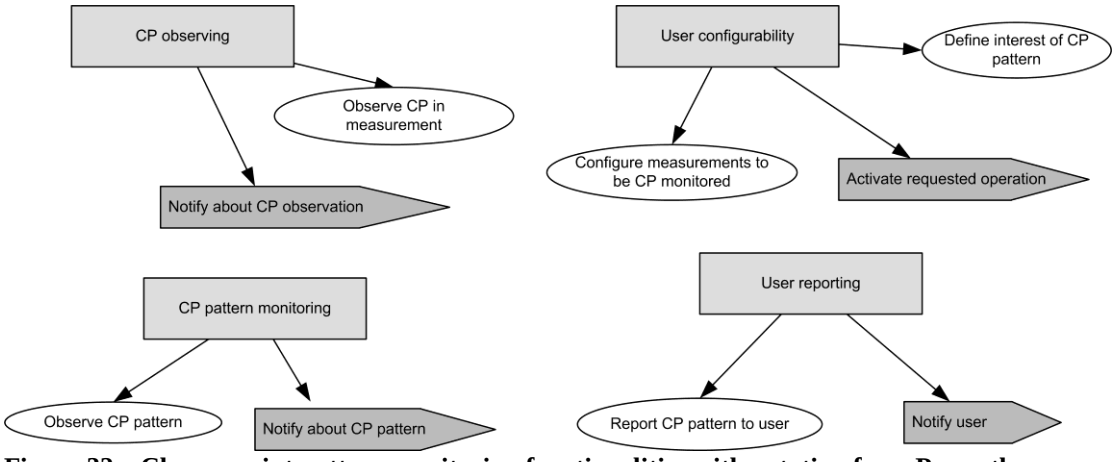

Figure 32 – Change point pattern monitoring functionalities with notation from Prometheus.

The next step is the scenario development, which describes how the goals and actions are linked together to form a normal operating sequence. Below is the *Change point pattern monitoring scenario,* which presents the normal operating sequence in which the user activates the pattern monitoring, change points and their patterns are observed, and finally pattern occurrence is reported to the user. This scenario is based on the typical use case of this experiment and its operation is triggered by the user.

Outline specification of Change point pattern observation scenario:

1. GOAL: Select measurements and setup pattern to be monitored
2. ACTION: Activate requested operation
3. GOAL: Observe CP in measurement
4. OTHER: wait for substantial change in measurement
5. PERCEPT: *CP observed*
6. ACTION: Notify about CP observation
7. GOAL: Observe CP pattern
8. OTHER: Check CP pattern occurrence
9. ACTION: Notify about CP pattern
10. GOAL: Report CP pattern to user
11. ACTION: Notify user

Discussion about the change point detection in the OTHER type step 4 can be found in chapter 6.5.3 and the step 8 is covered in the following chapter 6.6.3.The interaction of the system with the environment is the following. Perception is responsible for triggering activities in the case of *user activity* and when a *CP observed* is available in measurements. Internally, the system uses actions to trigger functions, and the only one that is visible outside the system is *Notify user*. Furthermore, the data used in this experiment are defined as the following.

Data stores for this experiment:

- *CP configuration and pattern setup* – contains information on what measurements should be observed for CPs and are available for pattern observation. Pattern setup contains definitions of CP patterns that are of interest to the user.
- *CPs* – contains records of online process change points and possibly stores some history of observed CPs.
- *Pattern monitoring data* – contains information about possible pattern findings and, when found, these should be informed to the user.
- *CP pattern report*s – records of user-specified patterns that have been found by the system.

After the system functionalities, operational sequences, and environment interaction are defined it is possible to group these elements into meaningful entities (agents). Grouping is an iterative process and the final version of this is illustrated in Figure 33. General agent roles specified in the architectural design (Chapter 4.4) are used to guideline the grouping.

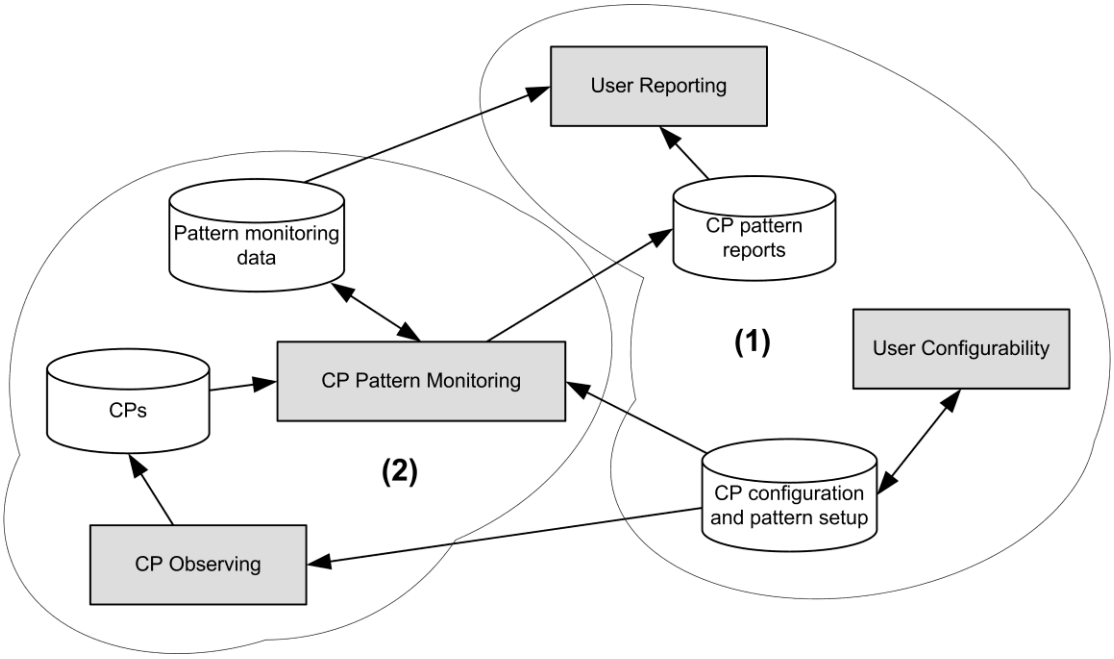

Figure 33 - Grouping of functionalities in change point pattern monitoring experiment with Prometheus notation. Clusters are named User agent (1) and Monitoring agent (2).

In the change point pattern monitoring experiment the grouping of functionalities produces the following clusters:

- **User agent:** The first cluster (1) holds the *User configurability* and *User Reporting* functionalities. The configuration functionality produces data for the *CP configuration and pattern setup* data store. The reporting functionality is responsible for reporting the patterns found in the system to the end user, on the event-based principle. Reporting is activated when new user reports are stored in the *CP pattern reports* data store. Nevertheless, if the user is interested in seeing what happens on the way, accessing *Pattern monitoring data* enables him/her to view online how the monitoring is advancing. As these events are rather rare in general, one agent of this type per user is enough. This cluster of functionalities is motivated by the *Client agent* architectural role.
- **Monitoring agent:** The second cluster (2) involves the actual monitoring functionalities, namely the *CP Observing* and *CP Pattern Monitoring* functionalities. These are operations that have to be active all the time so as to be able to observe changes in data. These functionalities get their initialisation from the configuration file, *CP configuration and pattern setup*. The number of these agents is not restricted, and it depends on the number of observed measurements and configured pattern observation. The architectural roles of *Information* and *Process agents* influence this clustering.

Closer examination of the clustering may raise the question of why the configuring and reporting functionalities are not separated, as was done in the previous experiment (Experiment 3) in a fairly similar situation. Keeping these two functionalities together is a design choice that has been made in this experiment because the reporting of patterns is a rather complex task and it needs an advanced user interface. Integrating configuring, supervising, and reporting activities into one specialised user interface makes this functionality more useful for the user.

Specifying interactions: Interaction protocols are developed on the basis of scenarios and the related agent descriptions. Change point pattern monitoring interactions are divided into two natural parts. The first of these is the initialisation interaction protocol, which is illustrated in Figure 34. The initialisation protocol shows how the responsible peers are first searched for from the directory and then these are requested to report about change point occurrences. The initialisation also shows the generation of the data structure.

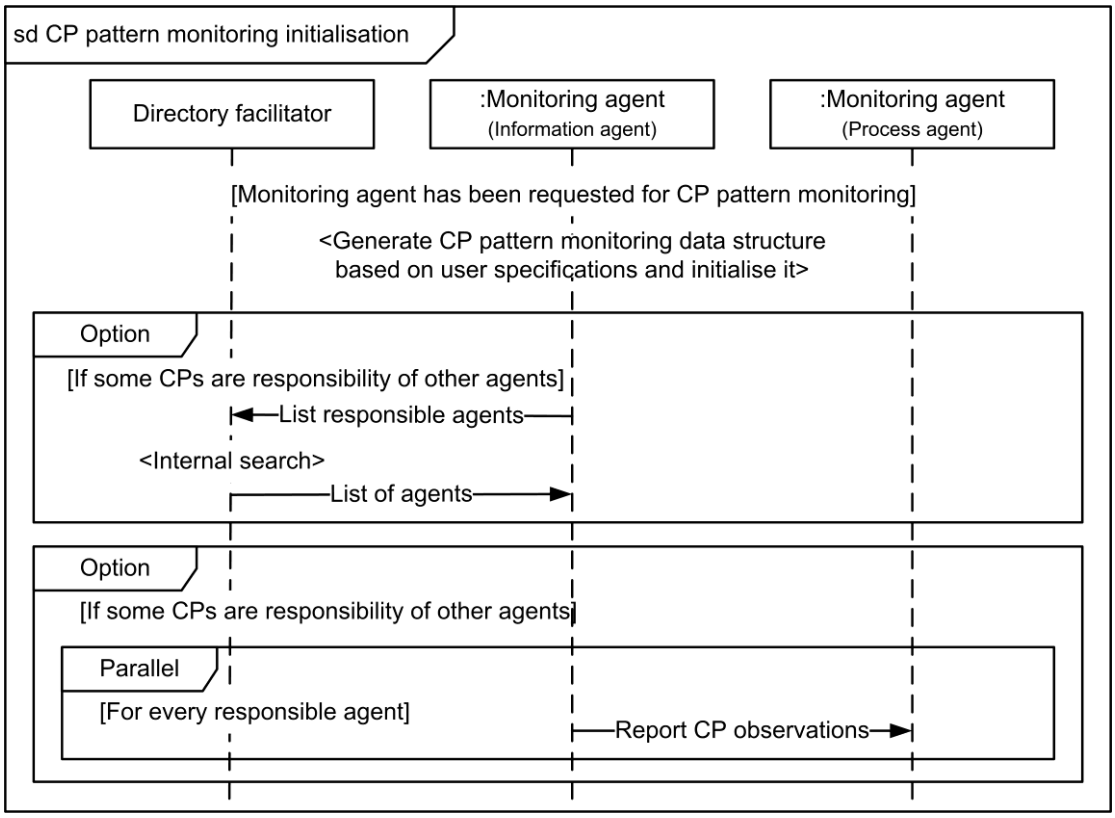

Figure 34 – CP pattern monitoring initialisation interaction protocol using Prometheus notation.

The second part of the change point pattern monitoring interaction protocol is responsible for the actual monitoring activity, and it is illustrated in Figure 35. This shows how change point occurrences are received and pattern occurrence is monitored by the information agent. If patterns have appeared then the user is notified. And finally, when the user deactivates the whole monitoring, the responsible agents are released.

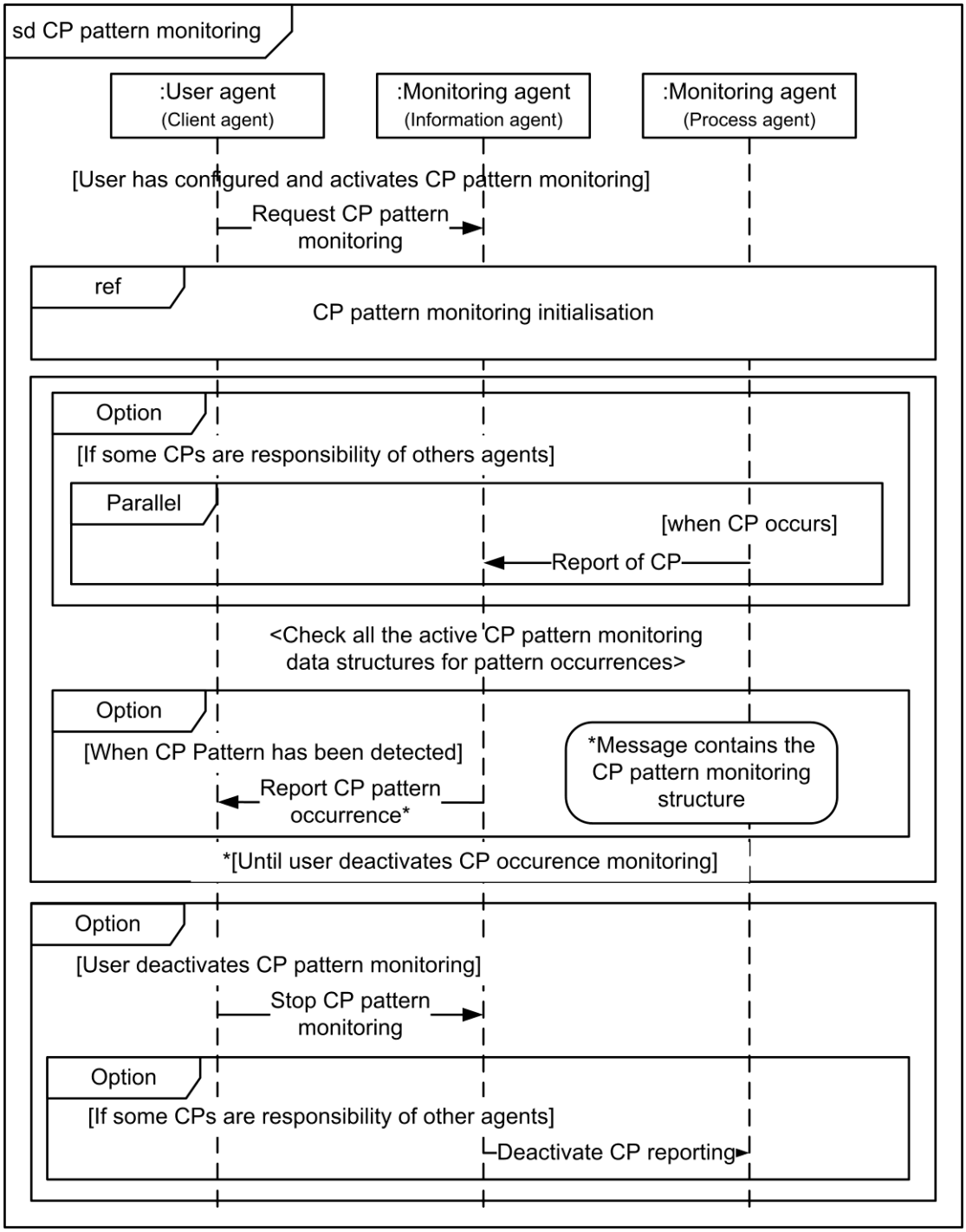

Figure 35 – CP pattern monitoring interaction protocol using Prometheus notation.

These interaction protocols together realise the main use case scenario presented in the introduction. In change point pattern monitoring roughly half of the operation is based on agent negotiations and the other half on the actual pattern monitoring.

6.6.3 Monitoring of change point patterns

This experiment uses Change Point (CP) detection as a method for generating events from time series data for inference making. Detecting and observing CPs was already discussed in the previous experiment and in Chapter 5.2.3. In this experiment, the user is interested in detecting event patterns in which the individual events are the respective CP observations. Because these CPs are events, instantaneous points in time, they do not have duration. This makes the patterns easier to configure, as there are not so many possible temporal relations between the two preceding events. Compared, for example, to the approach that was used by Lowe et al. (1999), which had 13 different relationships between two events, the pattern model promoted here also makes the detection simpler. Without the duration option the pattern definition consists of an event sequence specifying order and time constraints specified with *not before* and *not after* parameters. These properties are illustrated in Figure 36.

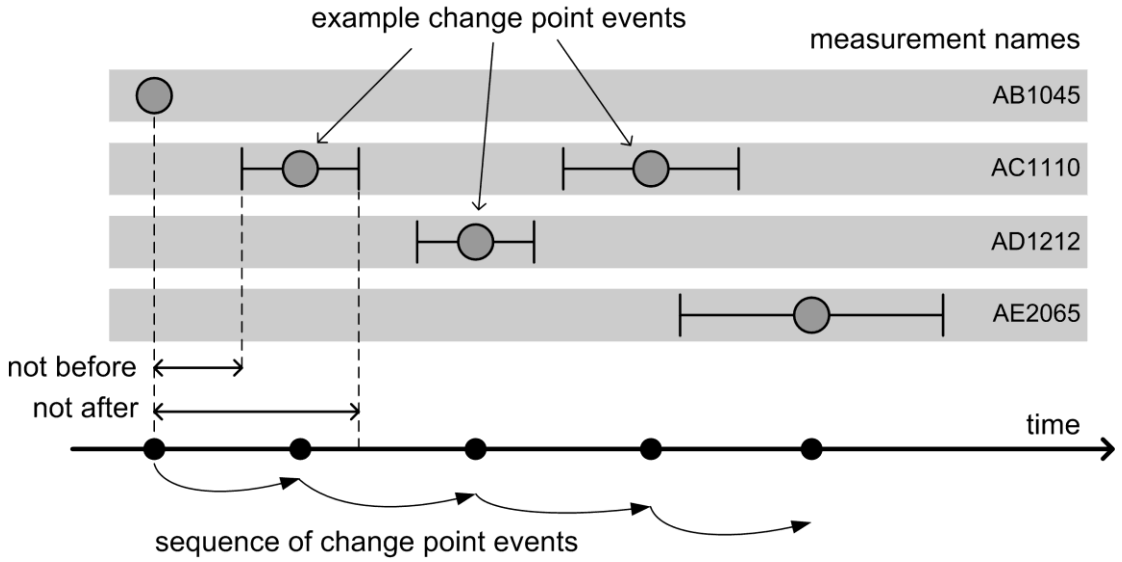

Figure 36 - Example change point pattern.

The illustrated pattern model is designed to be simple and understandable but it can easily be updated to a more advanced one in the future if needed. Table 10 shows the *CP configuration and pattern setup* data structure that is used to store the pattern monitoring task parameters, and is used to transmit the user-defined pattern monitoring task configuration to the monitoring agent.

Table 10 - CP configuration and pattern setup data structure.

Parameter name	Type	Example values
Pattern name	String	Pattern ABCD
Observed measurements	List of Strings (names)	AB1045, AC1110, AD1212, AE2065
CP pattern	Ordered list of; String (name), time (not before), time (not after)	AB1045, null, null, AC1110, 15min, 35min, AD1212, 20min, 40min, AC1110, 15min, 45min, AE2065, 5min, 50min

When the pattern monitoring task is activated, this configuration data structure is used to generate the *Pattern monitoring data structure*, shown in Table 11. In this structure, the *Observed CP events* is a copy of the *CP pattern table* with additional time stamp information. The time stamp is used to store the CP event occurrence information, which is used in the monitoring algorithm.

Table 11 - Pattern monitoring data structure.

Parameter name	Type	Example values
Pattern name	String	Pattern ABCD
Observed measurements	List of Strings (names)	AB1045, AC1110, AD1212, AE2065
Observed CP events	Ordered list of; String (name), timestamp (actual) time (not before), time (not after)	AB1045, 15:08, null, null, AC1110, 15:20, 15min, 35min, AD1212, 15:31, 20min, 40min, AC1110, null, 15min, 45min, AE2065, null, 5min, 50min

The pattern monitoring algorithm may be implemented to work in both directions; based on forward and backward scan principles. Computationally more efficient algorithms are based on the backward direction, e.g. the Boyer-Moore string-matching algorithm (Hume and Sunday 1991). However, the backward direction does not enable the user to see how things evolve in time step by step. Therefore, the forward direction should be used when applied to process monitoring to allow the user to diagnose the operation in an online manner. Support for tracing the monitoring as it advances may also be seen in agent design (Figure 28), in which *Pattern monitoring data* is connected to the *User reporting* functionality.

The pattern monitoring algorithm should be based on a step-by-step comparison of the defined patterns presently under observation and newly observed CP events that are received by the monitoring agent. In this experiment the CP pattern monitoring is based on the following principle (Figure 37):

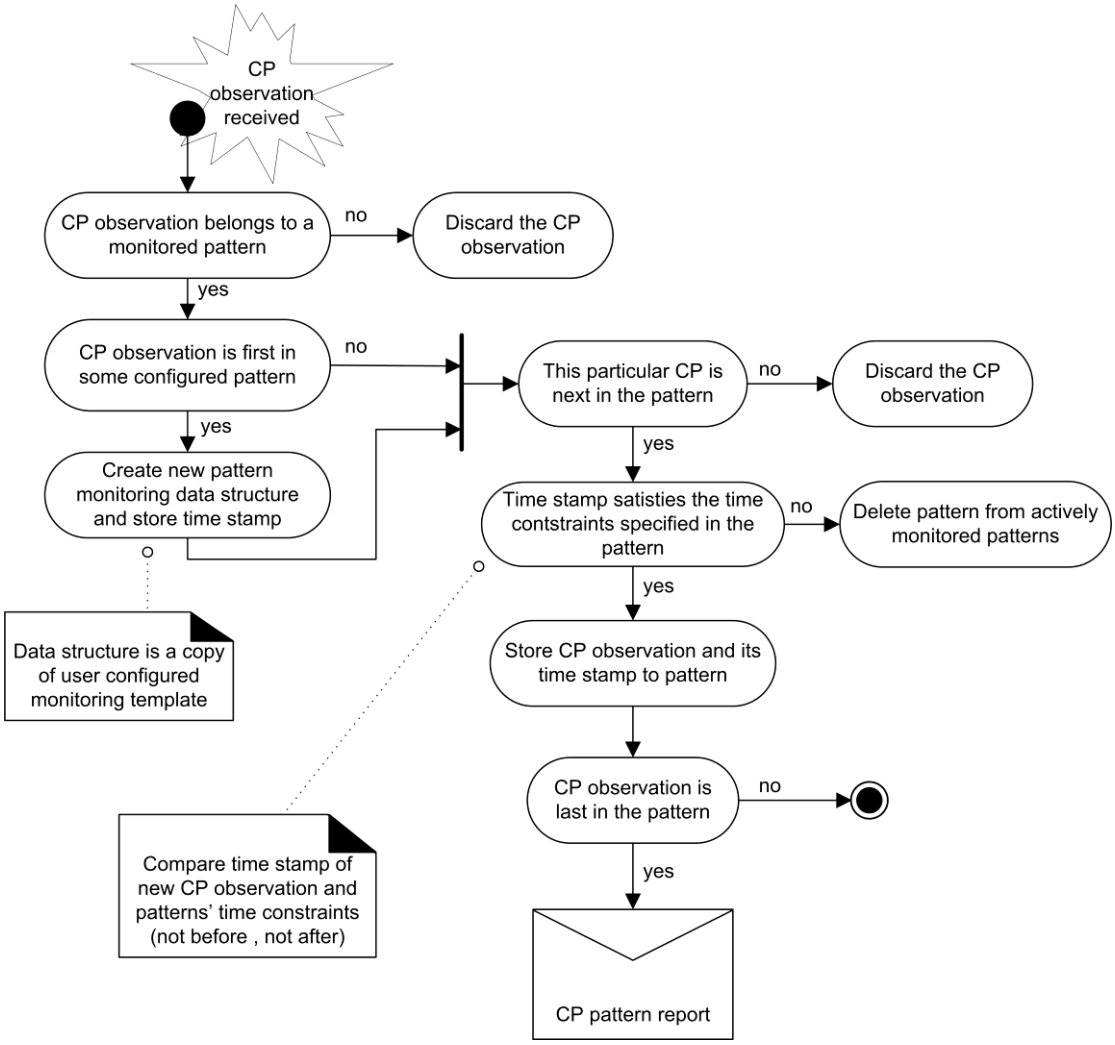

Figure 37 - Specification of CP pattern monitoring related activities, which shows in principle how monitoring agent processes newly received CP observation (using Prometheus notation).

This algorithm design does not take into account the possibility of parallelism, as there is no option to define the fact that two or more CPs may occur in mixed order. Support for this might be added to the algorithm in the future.

Finally, when the monitoring finds a matching pattern from the event flow it copies the current monitoring data to the report. Table 12 shows this reporting data structure, which is constructed when the pattern being monitored is found.

Table 12 - CP pattern reports data structure.

Parameter name	Type	Example values
Pattern name	String	Pattern ABCD
Observed CP events	Ordered list of; String (name), timestamp (actual) time (not before), time (not after)	AB1045, 10:08, null, AC1110, 10:21, 15min, 35min, AD1212, 10:44, 20min, 40min, AC1110, 11:14, 15min, 45min, AE2065, 11:25, 5min, 50min

6.6.4 Discussion

This experiment seems to benefit from an agent approach in its design. The functionality of the experiment may be designed and constructed relatively easily with an agent approach. The CP monitoring may be decentralised with an agent approach and the pattern monitoring may be utilised on an event-based principle. In this way the agent negotiations are fully utilised and data abstraction may be performed as near to the data source as possible. This experiment was the fourth to be designed and it used the same CP detection as Experiment 3. The benefits of this experiment functionality within industrial-scale process control are unknown, as this operation was not motivated by industrial partners. For this functionality to prove its industrial potential it should be tested in a real process.

This experiment was about detecting patterns in which individual events are change points, observed from time series data. This is a new functionality that is not currently available within commercial process automation systems. This functionality was thought to be most useful in the normal monitoring of processes, but it is probably also beneficial when monitoring the performance of preplanned tasks. Events other than CP-type ones could easily be added to the design of this experiment, and the functionality could be expanded that way. Such additions could be e.g. definitions of exact values that quantities are not supposed to exceed. Consideration of these other event generation principles was intentionally excluded from this design in order to keep the focus on the principal design issues.

6.7 Experiment summary and open questions

6.7.1 Summary of experiment results

The agent approach was used to develop four new different types of monitoring functions for process automation users. These experiments were developed in order to get an understanding about the feasibility of the agent approach, the design methodology used, and also of the system architecture developed for services of this type in process automation. As these experiments covered different aspects of process monitoring, they provided feedback on the design choices from multiple perspectives. Table 13 summarises the result of experiments and describes the functionalities that were added and used in the architecture.

Table 13 - Summary of results of experiments and functionalities needed in the architecture.

Experiment name and short description	Experiment results	Functionalities in the agent architecture
Experiment 1 *Temporal monitoring*	Tool for monitoring relational constraints between process quantities. Decentralised operation using flexible agent interactions, and the use of directory service to find information providers.	Modularisation of agent's internal operation, separating local monitoring (CSP engine) and agent interactions (handling the overall task). First use of monitoring as operating principle.
Experiment 2 *Search of process events*	Search tool for finding relations in process events and measurements. Merging of partial information and converting from one user group's viewpoint to other. Especially, operators' viewpoints on devices differ from maintenance personnel viewpoint.	Tool support for use of formal data models and common base ontology. Wrappers covering external data sources, conversion from one viewpoint to other, enabling e.g. legacy databases to be used.
Experiment 3 *CP density monitoring*	Process "fever" detection utilising CP occurrence monitoring. A new functionality making use of symbolisation of process measurements.	Refinement of agent internal modularised design, especially separation of numeric processing (with change point detection) and symbolic functionalities.
Experiment 4 *CP pattern monitoring*	CP pattern monitoring functionality. A new operational idea exploring agent interaction capabilities and data processing.	Further refinement of agent's internal modularised design as a result of developing CP pattern monitoring functionality. Outputs the final layered structure illustrated in Chapter 5.

In the table, the results of experiments present issues related to the general applicability of the functionality and are, as such, valuable to a wide audience range. The functionalities in the architecture are more about the technology itself and are thus more valuable to persons who are making decisions in the software area.

6.7.2 Conclusions and open questions

The experiments demonstrate four new functionalities to be used in process monitoring. These all support the flexible and user-configurable use of the monitoring functions, although the flexibility is restricted to cover only the designed functionality in these simplified cases. The first two experiments were motivated by industrial partners in our research project and are, as such, seen as functionalities with real potential for usage. The latter two were designed by the author, and these show what kind of monitoring functionalities could be offered with agent technology.

On the technical side the most interesting issue is the proposed benefits of goal-based operation offered by the BDI model. The literature suggests that goal-based operation is suitable for functionalities similar to the experiments that were conducted and running in environments comparable with process automation. Nevertheless, as may be seen in the agent design part of the experiments, the relatively simple and procedural operation does not require or benefit much from the use of goals. This aspect is especially visible in the scenario outlines, which shows that the goals and actions follow each other in a rather straightforward manner. This is most likely to be the result of simple operation, as in the scenarios there is no need to select what to do next. However, the outlines do not show the elapsed time. Because much of the monitoring functionality is about waiting for some specified event to occur, goal-based operation is still reasonable for realising the operation.

In addition, the outlined scenarios represent how agent operations interact and control the decision making activities. In scenarios OTHER and PERCEPT step types can be used to specify other than agent based functionalities. With the presented four scenarios the separation between agent functionality and decision making part was quite clear and natural. Therefore, from this perspective the used design methodology seemed to be suitable for the design of process automation related monitoring tasks. In real life applications the presented agent design methodology can be used in an early design phase to specify structure of the system and cover the distribution of operations. However, on the architectural level the role definitions of Information, Process, and Wrapper agent roles are mixed in each of the four experiments. This seems to be the result of unclear responsibility definition in the architectural level and also because the definition was defined to be related to the process setup and respective functionalities. Future research should address the role division issue.

Although all the experiments used the same ideas in the background, the architectural design, they were not running in exactly the same environment. This was because, between the experiments, the design phase and implementation was always partly started again from the beginning. The reason for this was that the experiments gave ideas as to how to further refine the design and implementation. This is the major issue that should be addressed when directing future research. It would be especially interesting to integrate all the demonstrated functionalities into one flexibly and easily configured environment and see how users in a real production context used and combined these in their monitoring work. However, this study did not aim to provide guidelines about what parts and properties of the monitoring task would be directly configured by the operator and what should be left to the system engineer. Therefore, a rather extensive study of user interface issues is needed before these functionalities and their combinations are on such a level that operators and users are capable of using them.

7 Discussion and conclusions

7.1 Discussion

The thesis is built on a holistic view. The approach is about combining multiple tools and methodologies together to form a new kind of system that facilitates flexible monitoring in process automation. In the background is the idea of utilising agent technology to extend the functionalities of process automation systems, and the aim of this thesis is to extend the agent technology research to cover monitoring functionalities.

Agents are designed to organise actions and operations in dynamic and distributed settings. Additionally, autonomy is argued to be a basic property of an agent, and it has been demonstrated to be suitable for the delegation of easy and laborious functionalities to machines. The academic research background provides sound theories for the utilisation of these aspects and lately systematic design methodologies have become available. Currently, one of the most viable approaches to the implementation of rationality seems to be the BDI (Belief-Desire-Intention) model, which has its basis in human thinking. In addition, a systematic step-by-step design methodology supporting BDI with enough quality was found to be available, and in the thesis the selected Agent-Oriented Software Engineering (AOSE) methodology was Prometheus.

The theoretically proven issues of agent technology seem to match the requirements set by monitoring in process automation. Other researchers have also found this (Theiss 2007; Wagner 2002) and demonstrations have been presented (Bunch et al. 2004; Buse and Wu 2007; Cockburn and Jennings 1995; Gentil 2006; McArthur et al. 2005; Salyda and Taylor 2007; Wörn et al. 2002). Although agents seem to be feasible, numerous aspects of it are still under heavy development and this practically postpones agent application in large-scale industrial projects. For example, FIPA organisation has IEEE accepted standards defining overall agent system structures and agent negotiations, but e.g. the Prometheus does not take FIPA issues into account in any of its design phases. Furthermore, practically all reported monitoring applications used distinct system structures and also implementation tools used varied.

Knowledge representation issues are central in organising rational operation with agents in a distributed manner, e.g. decision-making processes, but usable standards and tools for this are lacking within agent technology. At the same time knowledge representation issues within Semantic Web have much more interest and visibility in both academic research and industry than the respective agent community solutions. Therefore, in addition to agent technology, the Semantic Web and related tools were included in the system design because information-handling, partial data adaptation, data modelling, and structuring properties offered by it were seen as being usable in monitoring contexts. In addition, the development of Semantic Web has resulted in standards and tools that are highly usable. Nevertheless, it was found that the Semantic Web is also intended to cover control aspects related to information services, which, however, was found to be a rather underdeveloped subject.

Based on studies, agents and Semantic Web seemed to complete each other. And although the exploration of combining the best parts from both the Semantic Web and

agents has been started, the work is in a fairly embryonic stage (Dickinson and Wooldridge 2005; Greenwood et al. 2007; Huhns et al. 2005; Lassila 2007). Actually, there was already available an initial implementation of a system combining agents and the Semantic Web, but its development was halted before it was finalised (NUIN 2008). Furthermore, it was found that the OWL-S standard overlaps with the structures used in actual agent BDI model implementations; in particular, the plan context definition seems to be similar to the IOPE definitions in the OWL-S standard.

7.2 Conclusions

Technically process automation environment seems to enable the use of new software tools. With respect to studying the requirements of monitoring systems in process automation five desired properties were defined. (1) *Flexibility* is needed to be able to adapt to changing situations and (2) *delegation* to aid users to cope with the increasing amount of responsibilities. In addition, as important information is available in multiple systems around the organisation (3) *system integration* is needed to be supported. Finally, a system should support (4) *Knowledge handling* and be able to execute (5) *data processing*. Comparing these to the properties proposed for agent technology, the first three were found to be supported. Additionally, integration issues and knowledge handling were found to be supported by Semantic Web tools. Finally, data processing was determined to be covered by other solutions, e.g. statistical mathematic tools.

Based on the technical studies it was decided that the developed system should support both the BDI model provided by agent technology and also the use of Semantic Web tools. BDI model was selected because literature promotes its use for similar functionalities in other application areas. As no viable alternatives were found to be available, a tailored system was to be built to be able to study possibilities offered by agent technology. The design consisted of an architectural design and an internal structure, and the five above mentioned properties were used to direct the development process. In addition, available solutions were adapted as much as possible, e.g. the FIPA structure and interaction protocols.

As the design covers everything from the overall system structure to the internal design of an agent, it does not go deeply into details. Technically it is noted that a major part of agent technology and the Semantic Web focuses on processing symbolic or logical information. On the contrary, a major part of the important information in process automation is in numerical format, e.g. time series data. Although possible ways to link numerical and symbolic information together were proposed in this thesis, no general solution or algorithms were provided. Furthermore, the layered agent design aims to utilise simultaneously BDI model and Semantic Web tools but the thesis represents it as a proposal because the subject is generally unstable in the research community. On the other hand, the specified architectural design seems to be more stabile.

The thesis documents four industrially motivated experiments that were successfully realised with agent technology. The experiments were specified with a systematic AOSE methodology (Prometheus) and used guidelines defined in agent system architecture. Both of these aspects were found to be helpful and suitable for the experiments. On the basis of the experiments it is possible to state that the specified agent system architecture is general enough that a variety of monitoring

functionalities can be designed and implemented with it. The Prometheus agent design methodology was found to be useful for the *system specification* and *architectural design* phases, as these phases were applicable for the design of experiments. Furthermore, the step by step design procedure of Prometheus was suitable for figuring out issues related to monitoring tasks. However, it was found that Prometheus leaves much to be decided by the system designer, especially how to select goals to support rational operation in a distributed manner. Furthermore, the design of experiments revealed that the specified agent roles in architectural design were not yet strict enough, as information, process and wrapper agent roles were mixed.

The documented experiments demonstrate successfully that agent design covering structural and timing issues of operation can be separated from algorithm design. Although this was possible in the presented experiments it is unclear if this is true also more generally in monitoring. In addition, it may be stated that guidelines for selecting appropriate algorithms for decision-making would be beneficial. Nevertheless, the experiments failed to test and demonstrate how different decision-making modules and their results could be fused together. In addition, it may be stated that the relatively simple functionalities demonstrated in the experiments could have been realised without the BDI model, e.g. with procedural operation. This does not verify that a BDI model-based operation would not be useful when used in larger and more complex situations. However, in relation to monitoring functions in process automation the question about the benefits provided by the BDI model is left open.

All in all, this thesis shows that agent technology provides a natural and usable way to structure monitoring systems and the systematic agent design methodology may be exploited to develop monitoring functionalities in real industrial settings. In addition, an implementable system architecture using agent technology and Semantic Web was also successfully specified. Furthermore, new types of monitoring functionalities were successfully developed with the design methodology and presented agent architecture. The experiments described here are examples of functionalities that could be provided to monitoring users, and in the future these and a number of others should be gathered together to form a valuable toolkit for operators. These should be designed in such a way that the user is easily capable of restructuring and modifying them to automate the variety of operations that are needed to monitor processes successfully.

7.3 Future work

There are many possible directions for future research in the area of IT-supported monitoring in process automation. Although an agent-based system was used as the technical approach within this research, the ideas and possibilities are accessible with a multitude of software tools. On the one hand, the basic idea of providing the possibility of using user-configurable and predefined elements mixed together goes way beyond one specific implementation technique. Therefore, one possible future task could be to generalise the architectural design and functionalities of experiments that were conducted and implement these with different technology. On the other hand, it would be interesting to apply the system developed here to other problem domains, such as detecting weather fronts with monitoring agents from temperature, humidity, and air pressure measurement values.

One important aspect of further development could be process automation data models, which should be created in the future. Extensive models in the process automation area would offer the possibility of integrating information provided by individual functionalities. Quality models would also provide improved deduction properties and structurally more complex operations could be developed. The research in this has already started in many directions, e.g. in connection to the development of OPC UA, but a lot of work needs to be done before industrial applicability is gained.

In the long term new versions of technologies having similar properties to agents and Semantic Web will be available. When utilising these successfully in a process environment it will be possible to provide users a total monitoring solution with controlled mesh up of things that has great structural flexibility of handling modular information processing and accessing elements. The system would enable static information describing known aspects of process setup (general process structure, device properties and manuals, etc.) combined with dynamic information (values, trends, events, etc.) generated from physical processes to provide new functions to users. For example, navigation in process automation related information could and should be based on these both aspects. In addition, relevant information could be offered to human users similarly as news background stories. Partial technical solutions for all of these functionalities are already demonstrated or at least visible but systems combining them are not yet available.

References

ABB. 2007. IndustrialIT Extended Automation System 800xA. High integrity automation solutions for continuous productivity improvements. ABB Process Automation. System 800xA Brochure. document id: 3BUS092082R0101.

Appelqvist, P., Seilonen, I., Vainio, M., Halme, A. and Koskinen, K. 2002. Heterogeneous Agents Cooperating: Robots, Field Devices, and Process Automation Integrated with Agent Technology, 6th International Symposium on Distributed Autonomous Robotic Systems (DARS 2002). Fukuoka, Japan, June 25–27.

ARQ. 2008. ARQ query engine for Jena. Available at: http://jena.sourceforge.net/ARQ/.

AUML. 2008. An agent-based unified modelling language (AUML). Available at: http://www.auml.org.

Bann, J. J., Irisarri, G. D., Mohtari, S., Kirschen, D. S. and Miller, B. N. 1997. Integrating AI Applications in an Energy Management System. IEEE Expert: Intelligent Systems and Their Applications. Vol. 12, No. 6, pp. 53–59. DOI = http://dx.doi.org/10.1109/64.642962.

Barata, J., Camarinha-Matos, L. and Onori, M. 2005. A multiagent based control approach for evolvable assembly systems. 3rd IEEE International Conference on Industrial Informatics (INDIN '05), pp. 478–483. ISBN: 0-7803-9094-6.

Bayardo, R., Bohrer, W., Brice, R., Cichocki, A., Fowler, G., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A. and Woelk. D. 1997. Semantic Integration of Information in Open and Dynamic Environments. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 195–206.

Beckstein, C., Kraetzschmar, G., and Schneeberger, J. 1994. Distributed plan maintenance for scheduling and execution. Current Trends in AI Planning, EWSP'93. Second European Workshop on Planning, Frontiers in Artificial Intelligence and Applications Series, pp. 74–86.

Bergenti, F., Gleizes, M.-P., and Zambonelli, F. 2004. Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, Springer.

Berners-Lee, T, Hendler, J. and Lassila, O. 2001. The Semantic Web. Scientific American. Vol. 284, No. 5, pp. 34–43.

Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Gomez-Sanz, J., Leite, J., O'Hare, G., Pokahr, A. and Ricci. A. 2006. A Survey of Programming Languages and Platforms for Multi-Agent Systems. Informatica. Vol. 30, pp. 33–44.

Bordini, R. H., Dastani, M. and Winikoff, M. 2007. Current Issues in Multi-Agent Systems Development (Invited Paper). Post-proceedings of the Seventh Annual International Workshop on Engineering Societies in the Agents World. LNAI 4457, pp. 38–61.

Brennan, R. W., Fletcher, M. and Norrie, D. H. 2002. 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.

Bunch, L., Breedy, M., Bradshaw, J. M., Carvalho, M. and Suri, N. 2005. KARMEN: Multi-agent Monitoring and Notification for Complex Processes Holonic and Multi-Agent Systems for Manufacturing. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. Vol. 3593/2005. ISBN 978-3-540-28237-2.

Bunch, L., Breedy, M., Bradshaw, J. M., Carvalho, M., Suri, N., Uszok, A., Hansen, J., Pechoucek, M. and Marik, V. 2004. Software agents for process monitoring and notification. In Proceedings of the

2004 ACM Symposium on Applied Computing (Nicosia, Cyprus, March 14 - 17, 2004). SAC '04. ACM, New York, pp. 94–100. DOI= http://doi.acm.org/10.1145/967900.967921.

Buse, D. P. and Wu, Q.H. 2007. IP Network-based Multi-agent Systems for Industrial Automation Information Management, Condition Monitoring and Control of Power Systems. 187 p. Springer. ISBN: 978-1-84628-646-9.

Buse, D.P., Sun, P., Wu, Q.H. and Fitch, J. 2003. Agent-based substation automation. IEEE Power and Energy Magazine. Vol. 1, No. 2, pp. 50–55. ISSN: 1540-7977.

S. Bussmann and D.C. McFarlane. 1999. Rationales for Holonic Manufacturing Control. In Proc. of the 2nd Int. Workshop on Intelligent Manufacturing Systems. Leuven, Belgium, pp. 177–184.

Castro, J., Kolp, M. and Mylopoulos, J. 2001. A requirements-driven development methodology. In Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering, CAiSE'01, pp. 108–123.

Chiang, L.H., Russell, E.L. and Braatz, R.D. 2001. Fault Detection and Diagnosis in Industrial Systems. Springer. ISBN: 978-1-85233-327-0.

Chokshi, N. N. 2005. Holonic Process Control: A Distributed, Collaborative Approach to the Control of Chemical Process Operations, Doctorate thesis, University of Cambridge, Churchill College.

Chokshi, N. N. and 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. and McFarlane, D. 2008. A distributed architecture for reconfigurable control of continuous process operations. Journal of Intelligent Manufacturing. Vol. 19, No 2, pp. 215–232. Springer.

Chopin, N. 2007. Dynamic detection of change points in long time series. Annals of the Institute of Statistical Mathematics. Vol. 59, No. 2, pp. 349–366.

Ciocoiu M, Nau D. 2000. Ontology-based semantics. In: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, Breckenbridge, CO, April 12–16.

Cockburn, D. and 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.

Daw C. S., Finney C. E. A. and Tracy E. R. 2003. A Review of symbolic analysis of experimental data. Review of Scientific Instruments. Vol. 74, No. 2, pp. 915–930.

Decker, K. S., Lesser, V. R., Nagendra Prasad, M. V. and Wagner, T. 1995. MACRON: an architecture for multi-agent cooperative information gathering. In Proccedings of the CIKM-95 Workshop on Intelligent Information Agents.

Deen, S. M. 2003. Agent-Based Manufacturing—Advances in the Holonic Approach. Berlin, Germany. Springer-Verlag.

Dickinson, I. J. 2006. BDI Agents and the Semantic Web: Developing User-Facing Autonomous Applications. Ph.D. thesis. University of Liverpool.

Dickinson, I. and Wooldridge, M. 2005. Agents are not (just) web services: considering BDI agents and web services. In Proceedings of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005).

Eco, Umberto. 1998. Foucault's Pendulum. ISBN 0-436-14096-9.

EDDL. 2008. The Electronic Device Description Language (EDDL). Available at: http://www.eddl.org.

Elmusrati, M., Eriksson, L., Gribanova, K., Pohjola, M., Jäntti, R., Koivo, H., Johansson, M. and Zander, J. 2007. Wireless Automation: Opportunities and Challenges. Automation days '07. Helsinki. March 27–28. Suomen Automaatioseura ry, 8 pp.

Erl, T. 2005. Service-oriented Architecture: Concepts, Technology, And Design. 656 pages. Prentice Hall. ISBN 0131858580.

Farbrot, J.E., Bye, A. and Berg, Ø. 2000. How to design alarm systems that also work during plant upset conditions. Halden, 2000. (IFE/HR/E-2000/012) Annual Two-day Seminar and Workshop on Alarm Systems , 2, London.

FDI-Future. 2007. FDT Group and EDDL Cooperation Team Reach Agreement to Develop Unified Solution for Device Integration. Hanover, Germany. Available at: http://www.eddl.org/files/FDI-Future.doc.

FDT. 2008. Field Device Tool (FDT) Joint Interest Group.Available at: http://www.fdtgroup.org.

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

FIPA. 2002. FIPA Subscribe Interaction Protocol Specification. FIPA Standard SC00035H. Available at: http://www.fipa.org/specs/fipa00035/.

FIPA. 2008. The Foundation for Intelligent Physical Agents. Available at: http://www.fipa.org/.

Fischer, M. J. and Merritt, M. 2003. Appraising two decades of distributed computing theory research. Distributed Computing. Vol. 16, No. 2–3, pp. 239–247.

Gao, Y. and Kokossis, A. C. 2005. Agent-based intelligent system development for decision support in chemical process industry. European Symposium on Computer Aided Process Engineering. Puigjaner, L. and Espuña, A. (eds.). Elsevier Science B.V.

Gavrilova, T. and Laird D. 2005. Practical Design of Business Enterprise Ontologies. Proceedings of the 1st International IFIP/WG12.5 Working Conference on Industrial Applications of Semantic Web. Bramer, M. & Terziyan, V. (eds.) IFIP International Federation for Information Processing. Vol. 188. ISBN: 978-0-387-28568-9.

Gentil, S. 2006. Artificial Intelligence for Industrial Process Supervision. In Advances in Applied Artificial Intelligence. Ali, M. and Dapoigny R. (eds). LNAI 4031, Springer, pp. 2–11.

Georgoudakis, M., Kalogeras, A., Alexakos, C., Charatsis, K. and Koubias, S. 2005. A Holonic Ontology-Based Multi-Agent System for the Distributed Scheduling and Monitoring of Industrial Processes. 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005).

Gibbins, N., Harris, S. and Shadbolt, N. 2003. Agent-based Semantic Web Services. Journal of Web Semantics: Science, Services and Agents on the World Wide Web. Vol. 1, No. 2, pp. 141–154. ISSN 1570-8268.

Gilart-Iglesias, V., Macia-Perez, F., Mora-Gimeno, F.J. and Berna-Martinez, J.V. 2006. Normalization of Industrial Machinery with Embedded Devices and SOA. IEEE Conference on Emerging Technologies and Factory Automation (ETFA06). September 20–22, pp.173–180.

Gooijera, G. D. and Hyndmanb, R. J. 2006. 25 years of time series forecasting. International Journal of Forecasting. Vol. 22, No. 3, pp. 443–473.

Greenwood, D., Lyell, M. and Mallya, A. 2007. The IEEE FIPA approach to integrating software agents and web services. Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (IFAAMAS). Article No. 276. ISBN:978-81-904262-7-5.

Heck, B., Wills, L. and Vachtevanos, G. 2003. Software Technology for Implementing Reusable, Distributed Control Systems. IEEE Control Systems Magazine.

Helin, H. 2003. Supporting Nomadic Agent-based Applications in the FIPA Agent Architecture. Academic Dissertation. University of Helsinki, Department of Computer Science, Faculty of Science.

Hendler, J. A. 2006. Introducing the Future of AI. IEEE Intelligent Systems. Vol. 21, No. 3, pp. 2–4.

HoloMAS. 2000. First International Workshop on Industrial Applications of Holonic and Multi-Agent Systems. Available at: http://cyber.felk.cvut.cz/HoloMAS/2000/.

HoloMAS. 2001. Second International Workshop on Industrial Applications of Holonic and Multi-Agent Systems. Available at: http://cyber.felk.cvut.cz/HoloMAS/2001/.

HoloMAS. 2002. 3rd International Workshop on Industrial Applications of Holonic and Multi-Agent Systems. Available at: http://cyber.felk.cvut.cz/HoloMAS/2002/.

Honkanen, T. 2004. Modelling Industrial Maintenance Systems and the Effects of Automatic Condition Monitoring, Dr. Tech. Available at: http://lib.tkk.fi/Diss/2004/isbn9512268167/.

Huang, G. M. and Lei, J. 2007. A Semantic Based Software Architecture for Power Market Information Integration. Power Engineering Society General Meeting, IEEE. June 24–28, pp. 1–8. ISSN: 1932-5517. ISBN: 1-4244-1298-6.

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

Huhns, M. N., Singh, M. P., Burstein, M., Decker, K., Durfee, K. E., Finin, T., Gasser, T. L., Goradia, H., Jennings, P. N., Lakkaraju, K., Nakashima, H., Parunak, H. V. D., Rosenschein, J. S., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T. and Zavafa, L. 2005. Research directions for service-oriented multiagent systems. Internet Computing, IEEE. Vol. 9, No. 6, pp. 65–70.

Hume, A. and Sunday, D. 1991. Fast String Searching. Software – Practice and Experience. Vol. 21, No. 11, pp. 1221–1248.

Hyvönen, E., Viljanen, K., Tuominen, J. and Seppälä, K. 2008. Building a National Semantic Web Ontology and Ontology Service Infrastructure--The FinnONTO Approach. Proceedings of the European Semantic Web Conference (ESWC 2008). Springer.

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

Isermann, R. 2004. Model-based fault detection and dignosis – Status and application. The 16th IFAC Symposium on Automatic Control in Aerospace.

Jack. 2008. JACK™ is autonomous systems development platform. Available at: http://www.agent-software.com.au/products/jack.

JadeX. 2008. Jadex BDI Agent System. Available at: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/.

Jadex Web-bridge. 2008. Jadex Web-bridge Solution. Jadex Add-ons. Available at: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/addons.php.

Jammes, F. and Smit, H. 2005. Service-oriented paradigms in industrial automation. IEEE Transactions on Industrial Informatics. Vol. 1, No. 1, pp. 62–70.

Jason. 2008. Jason - a Java-based interpreter for an extended version of AgentSpeak. Available at: http://jason.sourceforge.net/JasonWebSite/.

Jena. 2008. Jena– A Semantic Web Framework for Java. Available at: http://jena.sourceforge.net/.

Jennings, N. R. 1999. Agent-based Computing: Promise and Perils. Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI). Stockholm, Sweden. (Computers and Thought award invited paper), pp. 1429–1436.

Jennings, N. R. 2000. On agent-based software engineering. Artificial Intelligence. No. 117, pp. 277–296.

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

Jennings, N. R., Sycara, K. P. and Wooldridge M. 1998. A Roadmap of Agent Research and Development. In Journal of Autonomous Agents and Multi-Agent Systems. Vol. 1, No. 1, pp. 7–36.

Jussila, L. 2006. Agent-Based Approach to Supervisory Information Services in Process Automation Master's Thesis, Helsinki University of Technology, Automation Technology Laboratory.

Jämsä-Jounela, S.-L. 2007. Future Trends in Process Automation. Annual Reviews on Control. Vol. 31, pp. 211-220.

Jämsä-Jounela, S.-L., Vermasvuori, M., Kämpe, J. and Koskela, K. 2005. Operator support system for pressure filters. Control Engineering Practice. Vol. 13, No. 10, pp. 1327–1337.

Kalogeras, A. P., Gialelis, J. V., Alexakos, C. E., Georgoudakis, M. J. and Koubias, S. A. 2006. Vertical integration of enterprise industrial systems utilizing web services. IEEE Transactions on Industrial Informatics. Vol. 2. No. 2, pp. 120–128.

Kaplan, S. 1984. The industrialisation of artificial intelligence: From by-line to bottom-line. The AI Magazine. Vol. 5, No. 2, pp. 51–52.

Keller, G. E. and Bryan, P. F. 2000. Process engineering: Moving in new directions, Chemical Engineering Progress 96(1): 41–50.

Klusch, M. 2001. Information agent technology for the Internet: a survey. Data & Knowledge Engineering. Vol. 36, No. 3, pp. 337–372.

Klusch, M., Omicini, A., Ossowski, S. and Laamanen, H. (eds.). 2003. Cooperative Information Agents VII. Proceedings 7th International Workshop, CIA 2003, Series: Lecture Notes in Computer Science. Vol. 2782. ISBN: 978-3-540-40798-0.

Kok, J. K., Warmer, C. J. and Kamphuis, I. G. 2005. PowerMatcher: multiagent control in the electricity infrastructure. Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS 2005), pp. 75–82. ISBN:1-59593-093-0.

Koskinen, T., Nieminen, M., Paunonen H. and Oksanen, J. 2003. The framework for indirect management features of process control user interfaces. 10th International Conference in Human - Computer Interaction. Greece. June 22–27.

Kuikka, S. 1999. A batch process management framework: domain-specific, design pattern and software component based approach. Dr. Tech. Thesis. VTT, Espoo. 215 p., VTT Publications : 398, ISBN 951-38-5541-4; 951-38-5542-2.

Kundzewicz, Z. W. and Robson A. (eds.). 2000. Detecting trend and other changes in hydrological data, World Climate Programme - Data and Monitoring, WMO, Geneva, pp. 113–119.

Laffey, T. J., Cox, P. A., Schmidt, J. L., Kao. S. M., and Read, J. Y. 1988. Real-Time Knowledge Based System. AI Magazine. Vol. 9, No. 1, pp. 27–45.

Laiho, E. 2005. Hälytysten hallinta Porvoon jalostamolla (In finnish). AEL/Insko seminaari. I801401/05 IX.

Laplante, P., Hoffman, R. R. and Klein, G. 2007. Antipatterns in the Creation of Intelligent Systems. IEEE Intelligent Systems. Vol. 22, No. 1, pp. 91–95.

Lassila, O. 2007. Programming Semantic Web Applications: A Synthesis of Knowledge Representation and Semi-Structured Data. Dr. Tech. Thesis. TKK Laboratory of Software Technology. ISBN 978-951-22-8984-4.

Last, M., Klein, Y. and Kandel, A. 2001. Knowledge discovery in time series databases. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics. Vol.31, No. 1, pp. 160–169. ISSN: 1083-4419.

Lastra, J. L. M. and Delamer, M. 2006. Semantic web services in factory automation: fundamental insights and research roadmap. IEEE Transactions on Industrial Informatics. Vol. 2, No. 1, pp. 1–11.

Laukkanen, I. 2008. Knowledge transfer and competence development in complex paper production environments. Dr. Tech. Thesis. Helsinki University of Technology. Faculty of Chemistry and Materials Sciences. Department of Biotechnology and Chemical Technology. Available at: http://lib.tkk.fi/Diss/2008/isbn9789512292141/.

Lavielle, M. and Lebarbier, E. 2001. An application of MCMC methods for the multiple change-points problem. Special section on Markov Chain Monte Carlo (MCMC) methods for signal processing. Signal Processing. Vol. 81, No. 1, pp. 39–53. ISSN:0165-1684.

Lewis, R. W. 1998. Programming Industrial Control Systems Using IEC 1131-3 (IEE Control Engineering Series).The Institute of Electrical Engineers. ISBN-10: 0852969503.

Lowe, A., Jones, R. W. and Harrison, M. J. 1999. Temporal Pattern Matching Using Fuzzy Templates. Journal of Intelligent Information Systems. Vol. 13, No. 1–2, pp. 27–45. ISSN: 0925-9902.

Luck, M., McBurney, P., Shehory, O. and Willmott, S. 2005. Agentlink roadmap: Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink. ISBN 085432 845 9.

Maes, P. 1994. Agents that reduce work and information overload. Communications of the ACM. Vol. 37, pp. 30–40.

Mangina, E. E., McArthur, S. D. J. and McDonald J. R. 2001. COMMAS (Condition Monitoring Multi-Agent System). Autonomous Agents and Multi-Agent Systems, No. 4, pp. 279–282.

Marík, V. and McFarlane, D. 2005. Industrial Adoption of Agent-Based Technologies. IEEE Intelligent Systems. Vol. 20, No. 1, pp. 27–35. ISSN:1541-1672.

Marík, V., Brennan, R. W. and Pechoucek, M. (eds.). 2005. Holonic and Multi-Agent Systems for Manufacturing, Second International Conference on Industrial Applications, of Holonic and Multi-Agent Systems, HoloMAS 2005. Copenhagen, Denmark. August 22–24. Proceedings. Lecture Notes in Computer Science 3593. Springer. ISBN 3-540-28237-8.

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

Marík, V., McFarlane, D. C. and Valckenaers, P. (eds.). 2003. Holonic and Multi-Agent Systems for Manufacturing, First International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2003. Prague, Czech Republic. September 1–3. Proceedings. Lecture Notes in Computer Science 2744. Springer. ISBN 3-540-40751-0.

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

Marík, V., Vyatkin, V. and Colombo, A. W. (eds.). 2007. Holonic and Multi-Agent Systems for Manufacturing, Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2007. Regensburg, Germany. September 3–5. Proceedings. Lecture Notes in Computer Science 4659. Springer. ISBN 978-3-540-74478-8.

Martin, D. and Domingue, J. 2007a. Semantic Web Services, Part 1. IEEE Intelligent Systems. Vol. 22, No.5, pp. 12–17.

Martin, D. and Domingue, J. 2007b. Semantic Web Services, Part 2. IEEE Intelligent Systems. Vol.22, No.6, pp. 8–15.

Mathieson, I., Sandy, D., Padgham, L., Gorman M., and Winikoff, M. 2004. An open meteorological alerting system: Issues and solutions. In Proceedings of the 27th Australasian Computer Science Conference, pp. 351–358.

Maturana, F. P., Staron, R. J., Tichý, P., Šlechta, P., and Vrba, P. 2005. A Strategy to Implement and Validate Industrial Applications of Holonic Systems. Lecture Notes in Computer Science 3593 Springer 2005, pp. 111–120. ISBN 3-540-28237-8.

McArthur, S. D. J., Booth, C. D., McDonald, J. R. and McFadyen, I. T. 2005. An agent-based anomaly detection architecture for condition monitoring. IEEE Transactions on Power Systems. Vol. 20, No. 4, pp. 1675–1682. ISSN: 0885-8950.

McClellan, M. 1997. Applying Manufacturing Execution Systems. St. Lucie Press, Florida. ISBN: 1574441353.

McFarlane, D. and 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.

McIlraith, S. A., Son, T. C. and Zeng, H. 2001. Semantic Web services. IEEE Intelligent Systems. Vol. 16, No. 2, pp. 46–53. ISSN: 1541-1672.

Metso. 2007. metsoDNA CR - a complete automation platform for better process results. Metso Automation Brochure.

Negroponte, N. 1995. Being digital. Coronet books. ISBN 0-340-64930-5.

Nodine, M., Ngu, A. H. H., Cassandra, A. and Bohrer, W. G. 2003. Scalable Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth™. IEEE Transactions on Knowledge and Data Engineering. Vol. 15, No. 5, pp. 1082–1098.

NUIN. 2008. nuin: a bdi agent framework for java. Available at: http://www.nuin.org/.

OASIS SOA. 2006. OASIS Reference Model for Service Oriented Architecture 1.0. Available at: http://docs.oasis-open.org/soa-rm/v1.0/.

OASIS UDDI. 2004. OASIS UDDI Version 3.0.2, Spec Technical Committee Draft, Dated 20041019. Available at: http://uddi.org/pubs/uddi_v3.htm.

OASIS WS-BPEL. 2007. OASIS Web Services Business Process Execution Language Version 2.0. OASIS Standard. 11 April 2007. Available at: http://docs.oasis-open.org/wsbpel/2.0/.

Obitko, M. and Marik, V. 2003. Adding OWL Semantics to ontologies used in multi-agent systems for manufacturing. First International Conference on Industrial Applications of Holonic and Multi-Agent systems (HoloMAS 2003).

Odell, J. 2002. Objects and agents compared. Journal of object technology. Vol. 1, No. 1, pp. 41–53. Available at: http://www.jot.fm/issues/issue_2002_05/column4/.

Olsson, G. and Piani, G. 1992. Computer Systems for Automation and Control. Prentice Hall. ISBN-10: 0134575814.

OPC. 2008. OPC Foundation. Available at: http://www.opcfoundation.org.

OPC UA. 2008. OPC Unified Architecture. Available at: http://www.opcfoundation.org/UA.

OWL. 2008. OWL Web Ontology Language Guide. W3C Recommendation 10 February 2004. Available at: http://www.w3.org/TR/owl-guide/.

OWL-S. 2008. OWL-S: Semantic Markup for Web Services. Available at: http://www.w3.org/Submission/OWL-S/.

OWL-S API. 2008. Java API for programmatic access to read, execute and write OWL-S service descriptions. Available at: http://www.mindswap.org/2004/owl-s/api/.

Padgham, L. and Winikoff, M. 2004. Developing Intelligent Agent Systems: A Practical Guide. 240 pages. John Wiley & Sons Ltd. ISBN: 978-0-470-86120-2.

Pakonen, A., Pirttioja, T., Seilonen, I. and Tommila, T. 2007. OWL Based Information Agent Services for Process Monitoring. 12th IEEE International Conference on Emerging Technologies and Factory Automation. Patras, Greece. September 25–28.

Parunak, H. V. D. 1997. “Go to the Ant”: Engineering Principles from Natural Multi Agent Systems. Annals of Operations Research. Vol. 75, pp. 69–10.

Parunak, H. V. D. 1999. Industrial and practical applications of DAI. In : Weiss, G. (ed.) Multiagent systems. MIT Press, pp. 377–421.

Paunonen, H. 1997. Roles of Informating Process Control Systems. Doctoral thesis.

Pechoucek, M. and Marik, V. 2008. Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies. International Journal on Autonomous Agents and Multi-Agent Systems. ISSN 1387-2532.

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

Pirttioja, T., Pakonen, A., Seilonen, I., Halme, A. and Koskinen, K. 2005. Multi-agent based information access services for condition monitoring in process automation. 3rd International IEEE Conference on Industrial Informatics (INDIN 2005). Perth, Australia. August 10–12.

Pirttioja, T., Seilonen, I., Appelqvist, P., Halme, A. and Koskinen, K. 2004. Agent-based architecture for information handling in automation systems. 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS’04).

Poutakidis, D., Padgham, L. and Winikoff, M. 2002. Debugging Multi-Agent Systems using Design Artifacts: The case of Interaction Protocols. The First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002). Italy. July 15–19.

Protégé. 2008. Protégé is a free, open source ontology editor and knowledge-base framework. Available at: http://protege.stanford.edu/.

Pyötsiä, J. 2005. ICT Opportunities and Challenges for Remote Services. Industrial Applications of Semantic Web. Bramer, M. & Terziyan, V. (eds.). IFIP International Federation for Information Processing, pp. 213–225. ISBN 978-0-387-28568-9.

Rao, A.S. and Georgeff, M.P. 1995. BDI agents: From theory to practice. Technical Note 56. Australian Artificial Intelligence Institute. Melbourne, Australia.

Rockwell. 2005. Combining Predictive, Preventive and Reactive Approaches turns maintenance into a Strategic. Rockwell ABJournal June 2005. Integrated predictive maintenance program.

Rockwell. 2006. Integrating Control and Information Systems. Rockwell Automation white paper. Available at: http://literature.rockwellautomation.com/, Pub. No. INF00-WP003A-EN-P.

Rockwell. 2007. Come together: IT- Controls Engineering Convergence Furthers Manufacturers' Success, Rockwell Automation white paper. Available at: http://literature.rockwellautomation.com/, Pub. No. CTITC-WP001A-EN-E.

Russell, S. and Norvig, P. 2003. Artificial Intelligence: A Modern Approach, Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey.

Sauter, T. 2005. Integration Aspects in Automation – a Technology Survey. 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Catania. September 19–22. Vol. 2, pp. 255–263.

Sayda, A. and Taylor, J. H. 2007. Toward A Practical Multi-agent System for Integrated Control and Asset Management of Petroleum Production Facilities. Proc. IEEE International Symposium on Intelligent Control.

Schiaffino, S. N. 2004. Personalization of User – Interface Agent Interaction. PhD Thesis, Universidad Nacional del Centro de la Provincia de Buenos Aires.

Schild, K. and Bussmann, S. 2007. Self-organization in manufacturing operations. Commun. ACM 50, 12 (Dec. 2007), pp. 74–79. DOI= http://doi.acm.org/10.1145/1323688.1323698.

Seilonen, I. 2006. An Extended Process Automation System: An Approach Based on a Multi-Agent System. Dr. Tech. Thesis. TKK Information and Computer Systems in Automation Laboratory.

Seilonen, I., Pirttioja, T., Pakonen, A., Halme, A. and Koskinen, K. 2006. Indirect Process Monitoring with Constraint Handling Agents. 4th International Conference on Industrial Informatics (INDIN 2006). Singapore. August 16–18.

Seppälä, J. and Salmenperä, M. 2005. Intelligent Visualisation of Process State Using Service Oriented Architechture. 16th IFAC World Congress. Prague, Czech Republic. July 4–8.

Shen, W. and Norrie, D. H. 1998. An Agent-Based Approach for Manufacturing Enterprise Integration and Supply Chain Management. Proceedings of the Tenth International IFIP WG5.2/WG5.3 Conference PROLAMAT 98. Tento, Italy. September 9–12, pp. 579–590.

Shen, W., Hao, Q., Yoon, H. Y. and Norrie, D. H. 2006. Applications of agent-based systems in intelligent manufacturing: An updated review. Advanced Engineering Informatics. Vol. 20, No. 4, pp. 415–431.

Sheridan, T. B. 1992. Telerobotics, Automation, and Human Supervisory Control. The MIT Press. Cambridge. Massachusetts. ISBN 0-262-19316-7.

Sierla, S. A., Christensen, J. H., Koskinen, K. O. and Peltola, J. P. 2007. Educational Approaches for the Industrial Acceptance of IEC 61499. 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2007). September 25–28.

SIRENA. 2008. Service Infrastructure for Real time Embedded Networked Applications. Available at: http://www.sirena-itea.org.

Soukup, J. and Soukup, M. 2007. The Inevitable Cycle: Graphical Tools and Programming Paradigms. Computer. Vol. 40, No. 8, pp. 24–30. ISSN 0018-9162.

SPARQL. 2008. W3C Recommendation for Query Language for RDF. Available at: http://www.w3.org/TR/rdf-sparql-query/.

Sycara, K., Giampapa, J. A., Langley, B. K. and Paolucci, M. 2003. The RETSINA MAS, a Case Study. in Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications. Garcia, A., Lucena, C., Zambonelli, F., Omici, A. and Castro, J. (eds.). Springer-Verlag, Berlin Heidelberg. Vol. LNCS 2603, pp. 232–250.

Takeuchi, J. and Yamanishi, K. 2006. A Unifying Framework for Detecting Outliers and Change Points from Time Series. IEEE Transactions on Knowledge and Data Engineering. Vol. 18, No. 4, pp. 482–492.

Tempich, C., Simperl, E., Luczak, M., Studer, R. and Pinto, H. S. 2007. Argumentation-Based Ontology Engineering. IEEE Intelligent Systems. Vol. 22, No. 6, pp. 52-59.

Tennenhouse, D. 2000. Proactive computing. Communications of the ACM. Vol. 43, pp. 43–50.

Terziyan, V. and Zharko, A. 2004. Semantic Web and Peer-to-Peer: Integration and Interoperability in Industry. International Journal of Computers, Systems and Signals. Vol. 4, No. 2, pp. 33–46. ISSN 1608-5655.

Theiss, S., Ploennigs, J., Vasyutynskyy, V., Naake, J. and Kabitzsch, K. 2007. Interactively configurable framework for industrial agents. 12th IEEE Conference on Emerging Technologies & Factory Automation (ETFA 2007). 25–28 Sept. 2007. pp. 384–391. ISBN: 978-1-4244-0826-9.

TII. 2008. IEEE Transactions on Industrial Informatics. Available at: http://ieee-ies.org/tii/.

Tommila, T., Ventä, O. and Koskinen, K. 2001. Next Generation Industrial Automation - Needs and Opportunities. VTT publication.

Tuomaala, M. 2007. Conceptual Approach to Process Integration Efficiency. Dr. Tech. Thesis. Helsinki University of Technology. ISBN 978-951-22-8788-8. Available at: http://lib.tkk.fi/Diss/2007/isbn9789512287888/.

Vasyutynskyy, V. & Kabitzsch, K. 2004. Architecture and Data Model for monitoring of Distributed automation systems. In TA 2004 - The 2nd IFAC Conference on Telematics Applications in Automation and Robotics, pp. 19– 24.

Venkatasubramanian, V., Rengaswamy, R. and Kavuri, S. N. 2003a. A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. Computers & Chemical Engineering. Vol. 27, No. 3, pp. 313–326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N. and Yin, K. 2003b. A review of process fault detection and diagnosis: Part III: Process history based methods. Computers & Chemical Engineering. Vol. 27, No. 3, pp. 327–346.

Venkatasubramanian, V., Rengaswamy, R., Yin, K. and Kavuri, S. N. 2003c. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. Computers & Chemical Engineering. Vol. 27, No. 3, pp. 293–311.

Ventä, O. 2005. Research View of Finnish Automation Industry, VTT publication.

Viinikkala, M., Syrjala, S. and Kuikka, S. 2006. A Maintenance Demand Analyzer – a Web Service based on a Semantic Plant Model. IEEE International Conference on Industrial Informatics, pp. 486–491. ISBN: 0-7803-9700-2.

Viroli, M., Holvoet, T., Ricci, A., Schelfthout, K. and Zambonelli, F. 2007. Infrastructures for the environment of multiagent systems, Journal of Autonomous Agents and Multi-Agent Systems. Vol. 14, pp. 49–60.

Vyatkin, V. V., Christensen, J. H. and Lastra, J. L. M. 2005. OOONEIDA: an open, object-oriented knowledge economy for intelligent industrial automation. IEEE Transactions on Industrial Informatics. Vol. 1, No. 1, pp. 4–17.

W3C. 2008. The World Wide Web Consortium (W3C). Available at: http://www.w3.org/.

W3C OWL-S. 2004. Semantic Markup for Web Services. W3C Member Submission 22 November 2004. Available at: http://www.w3.org/Submission/OWL-S/.

W3C SOAP. 2007. Version 1.2 Part 0: Primer (Second Edition), Nilo Mitra, Yves Lafon, Editors. World Wide Web Consortium, 27 April 2007. Available at: http://www.w3.org/TR/2007/REC-soap12-part0-20070427.

W3C SW. 2008. W3C Semantic Web Activity. Available at: http://www.w3.org/2001/sw/.

W3C SWSIG. 2008. Semantic Web Services Interest Group. Available at: http://www.w3.org/2002/ws/swsig/.

W3C WS. 2002. Web Services activity in The World Wide Web Consortium. Available at: http://www.w3.org/2002/ws/Activity.

W3C WS-ARCH. 2004. Web Services Architecture. Available at: http://www.w3.org/TR/ws-arch/.

W3C WS-CDL. 2005.Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation 9 November 2005. Available at: http://www.w3.org/TR/ws-cdl-10/.

W3C WSDL. 2007. Web Services Description Language (WSDL) Version 2.0. W3C Recommendation 26 June 2007. Available at: http://www.w3.org/TR/wsdl20/.

Wagner, T. 2002. An agent-oriented approach to industrial automation systems. Proceedings of the 3rd International Symposium on Multi-Agent Systems, Large Complex Systems and E-Business, pp. 508–524.

Wang, C., Ghenniwa, H., Shen, W. and Zhang, Y. 2004. Agent-Based Distributed Collaborative Monitoring and Maintenance in Manufacturing. Emerging Solutions for Future Manufacturing Systems. Camarinha-Matos, L. M. (ed.). IFIP TC5 / WG5.5 Sixth IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services. pp. 129–138.

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

Wong, H., Hu, B. Q., Ip, W. C. and Xia, J. 2006. Change-point analysis of hydrological time series using grey relational method. Journal of Hydrology. Vol. 324, pp. 323–338.

Wood, M. F. and DeLoach, S. A. 2000. An Overview of the Multiagent Systems Engineering Methodology. In Agent-Oriented Software Engineering – Proceedings of the First International Workshop on Agent-Oriented Software Engineering, 10th June 2000, Limerick, Ireland. Ciancarini, P. and Wooldridge, M. (eds.) Lecture Notes in Computer Science. Vol. 1957.

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

Wooldridge, M. J. and Jennings, N. R. 1999. Software engineering with agents: pitfalls and pratfalls. IEEE Internet Computing. pp. 20–27.

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

Yi-chuan, W., Yan-hui, G., Cong, W. and Xi-xin, Z. Distributed service management system: An agent and semantic web technology based approach. IEEE International Conference on Industrial Informatics (INDIN 2006). pp. 1063–1068. ISBN: 0-7803-9700-2.

Zambonelli, F., Jennings, N. R. and Wooldridge M. J. 2003. Developing Multiagent Systems: the Gaia Methodology, ACM Transactions on Software Engineering and Methodology, Vol. 12, No. 3, pp. 317–370.

HELSINKI UNIVERSITY OF TECHNOLOGY AUTOMATION TECHNOLOGY RESEARCH REPORTS

No. 18 Visala, A.,
Modeling of nonlinear processes using Wiener-NN representation and multiple models, November 1997.

No. 19 Xu, B.,
An interactive method for robot control and its application to deburring, November 1998.

No. 20 Zhang, X., Halme A.,
A biofilm reactor for a bacteria fuel cell system, August 1999.

No. 21 Vainio, M.,
Intelligence through interactions – Underwater robot society for distributed operations in closed aquatic environment, October 1999.

No. 22 Appelqvist, P.,
Mechatronics design of a robot society – A case study of minimalist underwater robots for distributed perception and task execution, November 2000.

No.23 Forsman, P.,
Three-dimensional localization and mapping of static environments by means of mobile perception, November 2001.

No.24 Selkäinaho, J.,
Adaptive autonomous navigation of mobile robots in unknown environments, December 2002.

No.25 Oksanen, T.,
Nelijalkainen nelipyöräinen robotti liikkuvana systeeminä, February 2003.

No.26 Suomela, J.,
From teleoperation to the cognitive human-robot interface, November 2004.

No.27 Aalto, H.,
Real-time receding horizon optimization of gas pipeline networks, April 2005.

No.28 Ylönen, S.,
Modularity in service robotics – Techno-economic justification through a case study, December 2006.

No.29 Appelqvist, A.,
Development of a biocatalytic fuel cell system for low-power electronic applications, December 2006.

No.30 Leppänen, I.,
Automatic Locomotion Mode Control of Wheel-Legged Robots, September 2007.

No.31 Oksanen, T.,
Path Planning Algorithms for Agricultural Field Machines, December 2007.

ISBN 978-951-22-9685-9 (print)
ISBN 978-951-22-9686-6 (pdf)
ISSN 0783-5477

Picaset Oy, Helsinki 2008