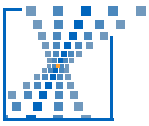
Resource Monitoring in Industrial Manufacturing Using Knowledge-Based Technologies

Lisa Theresa Abele



Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Fachgebiet für Geometrische Optimierung und Maschinelles Lernen

Resource Monitoring in Industrial Manufacturing Using Knowledge-Based Technologies

Lisa Theresa Abele

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende: Univ.-Prof. Dr.-Ing. Sandra Hirche

Prüfer der Dissertation:

1. Jun.-Prof. Dr. rer. nat. Martin Kleinsteuber
2. Univ.-Prof. Dr.-Ing. Alexander Fay, Helmut-Schmidt-Universität Hamburg

Die Dissertation wurde am 17.01.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 28.04.2014 angenommen.

Lisa Theresa Abele. *Resource Monitoring in Industrial Manufacturing Using Knowledge-Based Technologies.* Dissertation, Technische Universität München, Munich, Germany, 2014.

Abstract

Rising prices for natural resources, more severe legal regulations and an increased ecological awareness require means to improve the resource efficiency of industrial plants. An important means for improvement is to use a monitoring system to analyze the resource usage of plant elements. Based on the resource savings potential identified by such a monitoring system the plant engineering experts can optimize the plant's structure and processes accordingly.

Manufacturing trends impose increasing needs for changeable, application-independent monitoring systems that allow for the integration of knowledge on multiple plant aspects. Scientific research on automation systems showed that knowledge-based technologies are promising candidates for facing these challenges, but up to now there is no scientific investigation that proposes generic concepts for using knowledge-based technologies to monitor the resource usage of manufacturing plants. However, the specification of detailed concepts and strategies for implementing resource monitoring based on a knowledge-based technology stack is crucial in order to identify resource savings potential.

On this account, an approach for implementing a decentralized resource monitoring system by means of knowledge-based technologies is proposed within this thesis. The knowledge-based resource monitoring approach resides on a metamodeling architecture for integrating knowledge on a multitude of plant aspects. Furthermore, a hybrid combination of ontology and rule knowledge-representation paradigms is proposed for supporting the plant engineering experts in their tasks of engineering and maintaining the resource monitoring system. The architecture of the resource monitoring system is instantiated on a semantic technology stack to complete the assessment of the approach by evaluating the usability and scalability of the framework in different application scenarios.

Using the general concepts and methods of this thesis, the resource usage of manufacturing plants can be monitored and resource savings potentials can be identified. The chosen approach supports the collaborative integration of expert knowledge in a knowledge-based system, the computation of monitoring states for plant elements by means of rules, and the technical implementation for real-time application scenarios.

Acknowledgement

As every challenging project, this doctoral thesis would not have been possible without the inspiring advice of many people. The results of this thesis are based on interdisciplinary work during my three-year doctoral period at Fachgebiet für geometrische Optimierung und maschinelles Lernen (Technische Universität München) and Siemens Corporate Technology in Munich.

My special thanks go to my doctoral thesis supervisor Prof. Martin Kleinsteuber who gave me invaluable advice and the freedom to pursue my studies. On that account I would like to express my great appreciation to Prof. Alexander Fay for taking on the responsibility as second examiner, his interest in my research and valuable advice.

The opportunity to develop and implement my research in an industrial environment enriched the content of this thesis, but also enlarged my professional and personal experience. For giving me this possibility and also for the constant support throughout the entire project I would like to thank Michal Skubacz. Furthermore, I want to express my gratitude to Thorbjørn Hansen. He accompanied large parts of my research work as a supervisor in technical questions. We had many intensive discussions that helped me to gain insights in the area of knowledge modeling. I also want to thank Dr. Stephan Grimm for supporting me in the last part of my thesis elaboration. He gave thousands of helpful comments and ideas to early versions of this text and his help in finding the time to write it. Additionally, Dr. Sonja Zillner enabled fruitful discussions of my concepts and ideas benefiting from her profound and comprehensive knowledge.

I want to thank many colleagues and friends at Siemens AG and Technische Universität München for their feedback, their help and collaboration in realizing this work.

I particularly want to thank my parents who made all this possible and supported me all the time. My heartfelt thanks to Johannes for his permanent encouragement. Without him I wouldn't have had the courage to finish this work. I hope I will have the chance to pay back!

Lisa Abele, Munich, January 2014

Contents

Part I.

Foundations

1. Introduction

Huge amounts of various kinds of natural resources (e.g. electricity, oil, water) are used or emitted during manufacturing of products. In the past, high productivity of an industrial plant was much more important than an efficient use of resources, but the costs for resources are currently a huge part of the total production costs due to increasing costs for electrical energy and a shortage of resources. The European Parliament found that inefficient use of resources costs European industry around 630bn€per year [46]. Additionally, legal regulations require a limitation of resource consumption or emission, e.g. the German government aims to reduce CO_2 emission by 40% in 2020 [19]. Customers are further changing their purchasing behavior with regard to 'green' products [131]. Multiple government-funded research projects like RES-COM [15] or Green Carbody Technologies (GCT) [51] illustrate an increased need for giving answers to efficient resource usage in the manufacturing industry.

Towards this end, more advanced approaches need to be brought to the attention of managers and executives of manufacturing companies to increase the resource efficiency of their manufacturing plants. The first step towards an efficient usage of resources is to specify methods to understand the resource usage of the plants. Given that most complex machined parts (e.g. medical devices, turbines) require multiple production processes, an approach is needed to make plant engineers aware of the resource usage of their plants by monitoring resource-relevant parameters to optimize the plants' structure and processes accordingly.

Monitoring and analysis of resource usage of components and manufacturing processes is thus the first step towards increasing the resource efficiency of industrial plants. However, existing literature does not provide a holistic perspective on how to analyze the resource usage of manufacturing plants. Common types of monitoring systems in the manufacturing domain are: condition monitoring systems (used to monitor devices to schedule predictive maintenance), process monitoring systems (used to observe processes to ensure their correct execution), or energy monitoring systems (used to evaluate actual versus expected energy consumption). However, research on how to monitor resources (apart from energy) is still an open issue in the manufacturing domain.

1.1. Problem Statement

Similar to classic monitoring approaches (e.g. condition monitoring), the task of a resource monitoring system is to continuously analyze the resource usage of all plant elements such

as plant components or a production processes. Based on this analysis, the monitoring state of a plant element can be determined. The plant engineers should be able to optimize the plant's structure or processes by using the information specified within these monitoring states. Therefore, a resource monitoring system has to infer reliable monitoring states of plant elements to provide plant engineering experts with sufficient background information required to optimize the plant accordingly. To provide this background information different plant engineering aspects have to be considered: (i) structural plant aspects (such as resource flow between plant components) affect the monitoring results, (ii) a monitored process depends on a set of different kinds of resources, and (iii) a trade-off between different resource efficiency objectives depending on the current plant context (like an increase in energy efficiency or CO_2 emission reduction) should be reached. This requires more advanced models and algorithms allowing to insert additional plant knowledge, such as the current production process or structural plant aspects, in the monitoring system. Knowledge-based technologies allow for integrating different types of knowledge in a machine-readable manner and are, thus, a viable technology to implement such a monitoring system.

A new type of monitoring system named "Resource Monitoring System" is thus introduced:

Definition Resource Monitoring System: A resource monitoring system (RMS) is an information system used to observe the individual resource usage of the elements in an industrial plant, to analyze their resource usage and provide sufficient background information about multiple plant aspects, such as structural, process or context information.

1.1.1. Multitude of Plant Engineering Aspects

As pointed out in the previous section, an RMS needs to have knowledge about a multitude of plant engineering aspects to analyze the resource usage of plant elements. Figure 1.1 shows that these different plant aspects influence each other and are, thus, related to each other.

Decisions on these plant aspects made during the plant engineering process involve several engineering disciplines, e.g. mechanical, electrical or software engineering. Each discipline uses their own engineering tool and standards of the heterogeneous industrial tool landscape with often incompatible data formats [139]. The discipline-specific knowledge is usually stored in distinct partial plant models. Currently these partial plant models are handled separately as pointed out in [2]. A challenge for modern resource monitoring systems is to integrate these aspects in one single model to allow for an integrated view on all plant aspects.

Another challenge is that experts of the manufacturing domain have role-specific demands on a resource monitoring tool. These roles involve knowledge engineers, plant

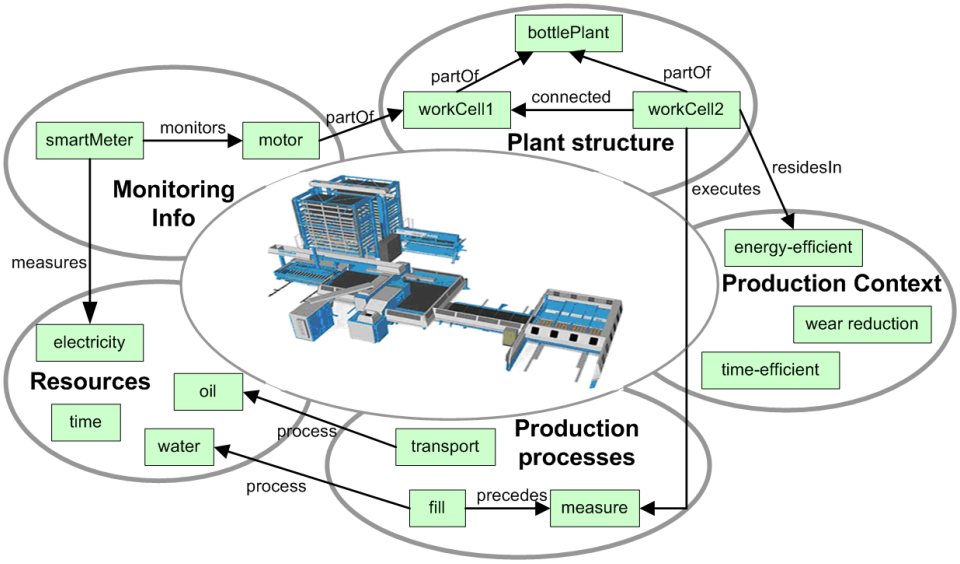


Figure 1.1.: A manufacturing plant is influenced by different aspects

engineering experts, third party suppliers, component manufacturers and plant operators. Current research usually focuses on defining partial plant models that correspond to the manufacturing goals from the perspective of one expert of a specific domain [160]. But to enable the development of an RMS in respondance to the demands of industrial practice of all domain experts, a joint consideration of the various perspectives is needed.

1.1.2. Implementation and Reconfiguration Issues

Current monitoring solutions are typically plant-specific software solutions and therefore require significant manual investment and effort for implementation. Once implemented, these systems have to be redesigned frequently because industrial plants regularly experience changes of design, technologies or business policies [44]. Monitoring systems should thus reuse thresholds, working conditions or events specified during the plant engineering process. The advantage of reusing such information is a decrease of manual effort for implementing a new system and for reconfiguring the monitoring system in case of changes.

Besides, plant components are usually executing subsequent steps of production processes and if an error occurs in the entire production process of a plant, this may lead to a combinatorial explosion in the number of monitoring states. It is not viable to explicitly program the computation of every monitoring state for individual process steps, instead logic-based mechanisms, such as rules, are a suitable tool to cope with this problem. Beyond that, rules are well-suited for an integration into knowledge bases and allow to save time and effort for implementing and adapting automation systems, such as monitoring

systems, as shown in [10, 78]. Even though the Automation of Automation (AoA) approach [119] tried to formalize the interchange formats for rules with regard to automation systems, there is not yet a standard rule format for this task [118].

As pointed out, knowledge-based models and rules allow to face the previously mentioned challenges, but the engineering of such knowledge-based models is usually a task of knowledge engineering experts since it requires specialized know-how. Usability of knowledge-based approaches, and especially Semantic Web technologies, has been an ongoing issue in the community [64]. Therefore, it is essential that a knowledge-based RMS is as easily usable and as highly self-descriptive as possible.

1.1.3. Decentralized System Architecture

This thesis was carried out in the context of the RES-COM project [15]. This research project is based on the paradigms of Industry 4.0 and aims to automatically conserve resources in industrial plants through active digital product memories (ADPMs) and context-aware embedded sensor-actuator systems. The new paradigm of Industry 4.0 suggests a change from the centralized to a decentralized architecture of automation systems [152]. The RMS built in the context of this thesis is based on such a decentralized architecture, where various Monitoring Units (MUs) communicate with each other and are able to monitor themselves by means of ADPMs. One of the major advantages of such a decentralized architecture using ADPMs is an improvement of the system implementation process due to several reasons: (1) manufacturers can produce intelligent components that can monitor themselves by providing extensive knowledge about their products, (2) an exchange of single components does not require modification of the entire system and downtime, (3) failure of one unit of the RMS will not affect the operation of the entire RMS.

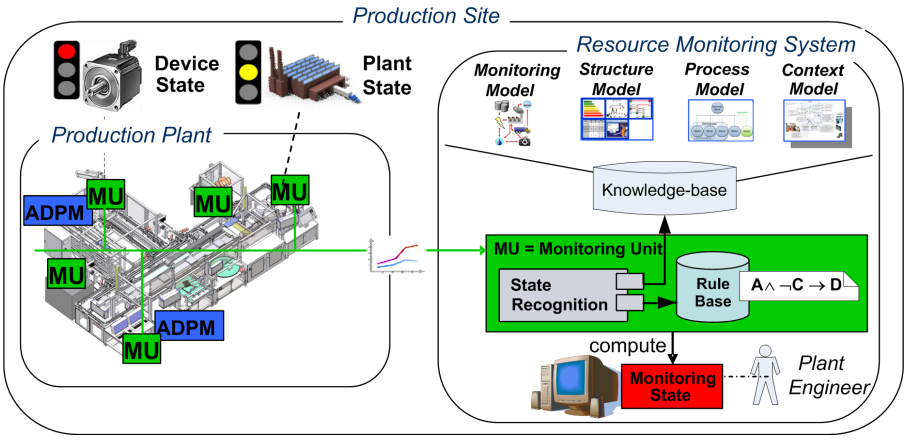


Figure 1.2.: Resource Monitoring System Architecture

The generic architecture of an RMS facing the challenges of the manufacturing domain

depicted in the previous subsections is presented in Figure 1.2. This figure shows a production plant equipped with several Monitoring Units based on ADPMs. These Monitoring Units compute monitoring states of devices or the entire plant by means of a state recognition module. The state recognition module has access to knowledge incorporated in different plant models (e.g. a structure model) stored in a knowledge base and uses a rule base to infer monitoring states. The computed states are then presented to the plant engineer via a user interface.

1.2. Research Approach

The objective of this work is to develop a decentralized resource monitoring system by means of knowledge-based technologies in order to analyze the resource usage of plant elements and infer monitoring states of plant elements. This objective is based on five research hypotheses presented in Section 1.2.1, which were derived from the current state of literature as presented in Chapter 3. These research hypotheses constitute the basis for the formulation of respective research questions in order to guide the validation of the hypotheses throughout the entire thesis. Furthermore, the particular contributions of this work are listed in Section 1.2.2 and the publications related to this work are sketched in Section 1.2.3.

1.2.1. Research Hypotheses and Questions

The first hypothesis deals with the task of defining a knowledge-based approach in alignment to the resource management factors of the industrial domain:

Research Hypothesis 1:
Economic, environmental and social pressures have increased the need for manufacturing companies to monitor their resource usage on a continual basis. Responding to these pressures the utilization purposes of resource-related information has to be clarified first. Consequently, the factors that influence the establishing of knowledge-based RMS need to be identified to respond to the demands of the manufacturing companies. A systematic knowledge engineering methodology for a knowledge-based RMS allows to address these factors.
Research Question 1:
What factors influence whether knowledge-based resource monitoring systems in manufacturing companies can be established and how can these factors be addressed by means of an appropriate methodology?

The second research hypothesis focuses on the generalization of a knowledge-based approach that integrates knowledge on different plant aspects:

Research Hypothesis 2:
Various branches of industry (e.g. metal industry, paper industry, steel industry) in different manufacturing engineering areas (e.g. process manufacturing, discrete manufacturing) need to monitor the resource usage of their plants. Multiple plant aspects, e.g. structural, process and context information, need to be integrated in one single model to compute reliable monitoring states. Application-independent knowledge-based techniques allow to integrate different plant aspects and can be applied in all branches and domains.
Research Question 2:
How can knowledge-based techniques integrate knowledge on different plant aspects to realize a resource monitoring system, which is universally applicable independent of the branch of industry and the manufacturing engineering area?

The third research hypothesis paves the way for a scalable system with sufficient performance for reasonably large industrial plants:

Research Hypothesis 3:
An increase of different scalability factors, such as the complexity of the plant or the sampling rate of sensors, influence the system performance and the scalability of knowledge-based systems. Due to such an increase, the management of plant knowledge and monitoring rules becomes difficult and the system performance of monitoring systems decreases. These issues can be addressed by a system architecture based on knowledge-based technologies.
Research Question 3:
What is the influence of an increase in different scalability factors on the performance and the scalability of a resource monitoring system realized by means of knowledge-based technologies?

The fourth research hypothesis focuses on usability aspects of the knowledge-based resource monitoring system:

Research Hypothesis 4:
The engineering of knowledge-based systems is usually a task for knowledge engineering experts, since it requires specialized know-how. This limitation impedes plant engineering experts to construct or adapt knowledge-based systems. Specific knowledge-based tools allow to improve ease-of-use and support plant engineers during the engineering process of a resource monitoring system.
Research Question 4:
What tools and procedures can be used to meet the RMS requirements and to

improve ease-of-use through knowledge-based techniques to support plant engineering experts during the engineering process of a Resource Monitoring System?

The fifth research hypothesis relates to the flexible manufacturing environment which requires a high reconfigurability of modern automation systems:

Research Hypothesis 5:
Modern industrial plants are a fast changing environment that requires steady adaptations. Knowledge-based techniques allow for decreasing the effort for the plant engineering experts to reconfigure an RMS in case of changes in the plant environment.
Research Question 5:
How can knowledge-based techniques be used to improve reconfigurability of resource monitoring systems?

1.2.2. Research Contributions

This research is based on insights gained from a research project in the manufacturing domain as well as a structured analysis of theoretical approaches. The contributions of this work span different fields of research and aim towards the realization of a decentralized resource monitoring system for the manufacturing domain. To this end, an applicability of the results shall be ensured.

For the first time, we propose a framework that enables a monitoring of resources in industrial plants in a decentralized manner. In contrast to other approaches, this framework supports the usage of a multitude of plant engineering aspects for monitoring. Another difference to current approaches is that the RMS can be implemented and reconfigured with a low effort by offering application-independence and a high degree of reusability.

A list of particular contributions of this research is sketched in the following:

- An empirical study was conducted to identify Resource Management Factors crucial for responding to the needs of industrial practice. Based on the results of the empirical studies, requirements for RMSs in the manufacturing domain were specified.
- A conceptual modeling framework is provided in this thesis as theoretical basis for an application-independent description of plant knowledge. By this, the theory-oriented work on metamodeling architectures is bridged with Semantic Web technology and industrial modeling paradigms. Furthermore, a hybrid combination of ontology and rule knowledge-representation paradigms, in which ontologies are used to organize a set of monitoring rules by structuring and reasoning about their constituents, is proposed.

- An implementation of the modeling framework on a semantic technology stack is proposed. This implementation fulfills the identified requirements for RMSs. The conceptual monitoring framework is realized by a plant model repository based on Semantic Mediawiki and underlying Semantic Web technologies. A reasoning approach is deployed for the task of supporting the plant engineering experts in engineering and maintaining the monitoring rules.

- A prototype of the system was tested in different application scenarios. The results of a usability and scalability evaluation of the prototypical system implementation are finally discussed.

This research work proposes a modeling approach for resource monitoring systems and thus does not provide methods to automatically interfere in control systems for optimization purposes, unlike various other approaches from related research, which often combine monitoring and control systems in a unique framework. Nevertheless, the provided approach can be used as a basis for defining other manufacturing automation systems, such as knowledge-based diagnostics or prognostics systems.

Seen from the industrial perspective, this thesis is a first step to realize RMSs in manufacturing plants in order to analyze the resource usage of the plants and identify resource savings potentials. To guarantee a broad acceptance of the RMS in the manufacturing environment, the collaborative integration of expert knowledge in a knowledge-based system is supported and the technical implementations for real-time application scenarios are specified. However, the prototype was only implemented on plants of smaller scale and needs to be tested on real-life examples by integrating it with a CAD system. Such an integration with current systems would offer a strong mechanism for computing semantically annotated resource monitoring states for manufacturing plants.

1.2.3. Publications

During my PhD studies, I have published the following articles in journals, international conferences or workshops:

- L. Abele, M. Kleinsteuber, and T. Hansen, “Resource Monitoring in Industrial Production with Knowledge-Based Models and Rules.” in *PIKM - the Workshop for PhD students at International Conference on Information and Knowledge Management (CIKM)*, Glasgow, Scotland, October 2011.

- L. Abele, L. Ollinger, I. Heck, and M. Kleinsteuber, “A Decentralized Resource Monitoring System Using Structural , Context and Process Information,” in *Trends in Intelligent Robotics, Automation and Manufacturing*, Kuala Lumpur, Malaysia, p. 371-378, Vol. 330, Springer Berlin - Heidelberg, November 2012.

- L. Abele, M. Anic, T. Gutmann, J. Folmer, M. Kleinsteuber, and B. Vogel-Heuser, “Combining Knowledge Modeling and Machine Learning for Alarm Root Cause Analysis,” in *IFAC Conference on Manufacturing, Modeling and Control (MIM13)*, St. Petersburg, Russia, June 2013.
- L. Abele, T. Hansen, and M. Kleinsteuber, “A knowledge engineering methodology for resource monitoring in the industrial domain,” in *IFAC Conference on Manufacturing, Modeling and Control (MIM13)*, St. Petersburg, Russia, June 2013. 2013.
- L. Abele, C. Legat, S. Grimm, and A. W. Müller, “Ontology-based Validation of Plant Models,” in *IEEE International Conference on Industrial Informatics*, Bochum, Germany, July 2013.
- L. Abele, S. Grimm, S. Zillner, and M. Kleinsteuber, “An Ontology-Based Approach for Decentralized Monitoring And Diagnostics,” currently reviewed by *Journal of Intelligent Manufacturing*.
- L. Abele, S. Grimm, M. Watzke, and M. Kleinsteuber, “Ontology-based Maintenance Support for Rule-based Monitoring of Industrial Plants,” currently reviewed by *Knowledge-Based Systems*.
- S. Zillner, A. Ebel, S. Martin, L. Abele, M. Gael, and N. Goldenberg, “A semantic modeling approach for the steel production domain,” in Proceedings of European Steel Technology & Application Days, to be published in march 2014.

Furthermore, I contributed to three patent applications:

- S. Grimm, L. Abele, A. Müller, and C. Legat, “Knowledge-Based Validation of Engineering Data,” submitted 2013-11-07, national file, application number: E2551DE00.
- L. Abele, S. Grimm, and S. Zillner, “System and Method for Handling Plant Engineering Data,” submitted 2013-09-19, international file, application number: 2013P06342US.
- L. Abele, S. Grimm, and M. Watzke, “Ontology-based Maintenance Support for Monitoring and Diagnostic Rules,” submitted 2013-11-15, national file, application number: E2594DE00.

1.3. Research Methodology and Outline

The following section gives an overview of the applied methodological procedure in this thesis. The spiral model of design research of [43] has been used as basis for the methodological framework of this thesis. This research methodology provides a framework, applicable for different forms of design research, especially for those that focus on

applied design research and a general understanding of design. The main reason why this methodology has been chosen was that it addresses ways to integrate a large number of small-scale research problems. This is particularly important for the aim of this research, since it is strongly related to solving multiple research problems as stated in Section 1.2.1. Furthermore, the spiral model benefits to researchers facing the academic need to produce reportable results, and the industrial need for powerful, reliable, validated tools and techniques. The designers of the spiral model of design research emphasize the importance to evaluate the research findings at each stage to assess what sort of foundation the research gives for work that depends on it. The chapters of part II-VI of this thesis described in the following were thus defined in alignment to the spiral model of design research, while the research questions defined in Section 1.2.1 were evaluated after each part as depicted in Figure 1.3.

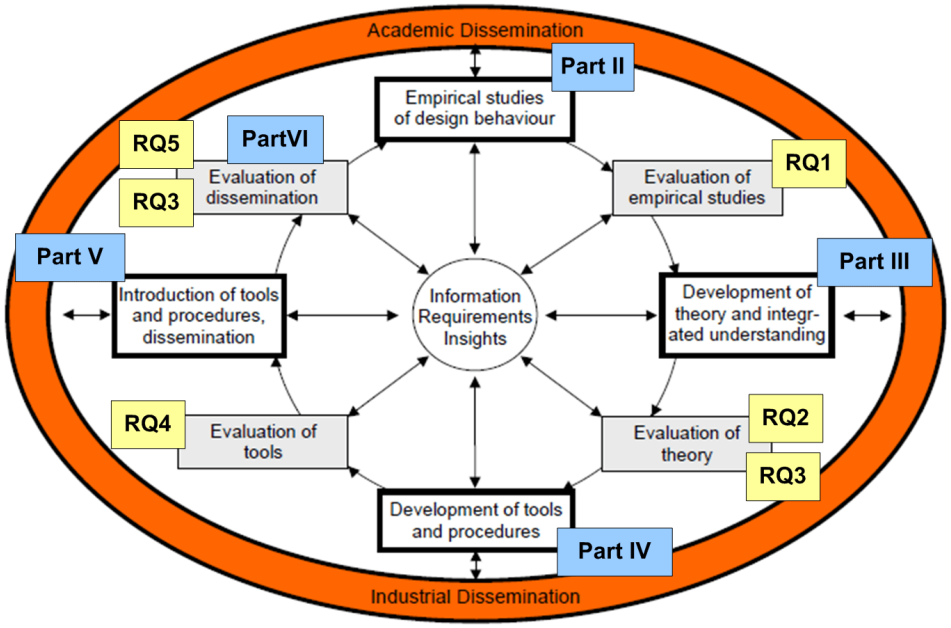


Figure 1.3.: Spiral Model of Research according to [43]

Part I - Foundations: In the first part, the foundations for this work on knowledge-based RMSs are layed out, presenting preliminaries in the fields of knowledge-based automation systems. In Chapter 2, the fundamental concepts of this thesis, the field of industry automation, related industrial automation systems and relevant representation languages and paradigms are presented. Chapter 3 discusses approaches of research fields related to this thesis.

Part II - Resource Management Factors: In the second part, Chapter 4 introduces the

empirical basis for the analysis of Resource Management Factors that knowledge-based RMSs for industrial plants have to face. The findings aim to ensure the practical impact of the research scope and allow to specify requirements for applicability of theory and procedures to be defined by providing an answer to RQ1. The results are evaluated regarding their generalisability by comparing them to results from other empirical studies in Chapter 5. To address the identified Resource Management Factors, a knowledge engineering methodology is developed in Chapter 6. Finally, a comprehensive list of requirements necessary for defining an RMS in consideration of the identified Resource Management Factors is depicted in Chapter 7.

Part III - Knowledge-Based Resource Monitoring Approach: In this part, a conceptual modeling approach is presented as theoretical basis for the RMS in Chapter 8. This approach provides the basis for integrating knowledge on different plant aspects in the RMS in a structured way. The application-independence of the conceptual modeling approach is evaluated by applying it on use cases of the steel manufacturing domain in Chapter 9. Chapter 10 discusses the difficulties in establishing methods to analyze the resource usage in modern manufacturing environments and specifies a monitoring model for integrating plant knowledge and rules in the RMS. Both the conceptual modeling approach and the monitoring model are evaluated with respect to their scalability in Chapter 11.

Part VI - Implementation Aspects: The purpose of this part is to describe important implementation aspects of the previously defined approach. Chapter 12 defines how the knowledge-based RMS is realized in order to meet the requirements specified in Chapter 7. To this end, a decentralized system architecture is proposed and implemented on a semantic technology stack. An important part of the evaluation is to compare the proposed technology stack with alternative technologies with respect to the requirements in Chapter 13. Finally, results of a usability study are discussed in Chapter 14.

Part V - Application Scenarios: In the fifth part, Chapter 15 describes several application scenarios for resource monitoring implemented in realistic plant environments. The performance of the RMS within the application scenarios is further evaluated and discussed in Chapter 16. Results of an evaluation of the reconfigurability of the RMS are provided in Chapter 17.

Part VI - Finale: In the final part, the thesis is concluded by summarizing the results of the evaluation questions in Chapter 18. Chapter 19 contains concluding remarks and an outlook on future research topics.

2. Fundamentals

This chapter introduces and defines the formal basis for this thesis. Altogether, three clusters of terms are introduced that were selected in consideration of the research areas affecting this work. The first cluster defines important concepts needed for understanding the knowledge-based resource monitoring established in this thesis in Section 2.1. Section 2.2 clusters terms of the industry automation domain and Section 2.3 presents related representation languages and paradigms.

2.1. Concepts for Knowledge-based Resource Monitoring

The most important concepts used within this thesis are defined in this cluster. The primary aim of this work is to define a **resource monitoring system** that builds on **partial plant models** for representing various aspects of knowledge about industrial plants. These models can be used in addition to sensor and process data to perform reasoning and infer sufficient knowledge to compute **monitoring states** for the plant and all relevant plant elements. The **monitoring rules** needed for the reasoning task are one fundamental part of this knowledge.

2.1.1. Resource Monitoring System

As stated in Chapter 1, an RMS is used to observe the individual resource usage of the elements in an industrial plant to analyze their resource usage and provide sufficient background information about multiple plant aspects.

To define an RMS, a taxonomy of resources that occur in a manufacturing environment was specified as shown in Figure 2.1. This taxonomy was defined based on classifications proposed in literature such as [25], where a distinction between “immaterial” (e.g. time) and “natural resources” (e.g. electrical energy) is made. Furthermore, an additional category ***Plant Resource*** is introduced for the manufacturing environment, which refers to specific resources that appear in industrial plants. One type of Plant Resources is ***Waste Material***, which encompasses unused or rejected material in a plant such as deficient products and unrectifiable rejects. Another type of Plant Resource is ***Component Wear***, which occurs when components installed within the plant brake due to deterioration or erosion.

Several resource usage aspects have to be considered when defining an RMS for a modern manufacturing environment. These aspects increase the complexity of RMS and involve resource efficiency assessment, measurement of resource usage and assignment of resources to component types.

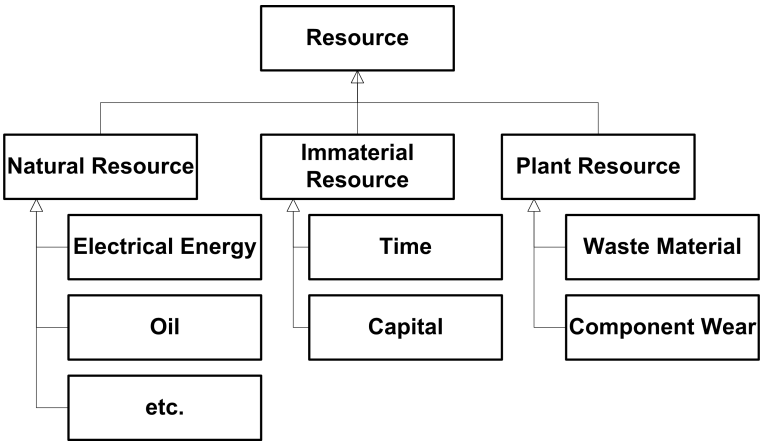


Figure 2.1.: Taxonomy of resources in the manufacturing environment

The first resource usage aspect concerns the assessment of the “natural resources” in industrial plants. An important natural resource, which is often considered in literature, is electrical energy. As stated in [40], key figures which are currently used for energy efficiency assessment, e.g. in the International Performance Measurement and Verification Protocol (IPMVP), are not detailed enough for production plants. These research results suggest that all secondary energy expenses have to be taken into account for monitoring the energy efficiency of a production plant. This includes energy consumption of all machine components and supply units, power drain during idle times, CO_2 emission as well as other sources of energy such as pressured air, hydraulics, extracted air and coolants. For this reason, an RMS should not focus mainly on energy monitoring, but more on the monitoring of secondary energy sources processed in a plant.

A second aspect is the absence of measurement equipment for the monitoring of resources. Currently, most of the industrial production plants are not monitoring their resource usage even if all the information necessary to measure the resource consumption were available in their control systems [144]. This information is typically not registered constantly but only in case of failure. An exception are steel manufacturing facilities where multiple parameters related to resource usage are continuously stored. As a result, additional sensors need to be installed in industrial plants to guarantee a reliable monitoring of the resource usage, such as acceleration sensors, flow rate sensors, ultrasonic sensors, etc. For reducing component waste, sensors that constantly the component’s deterioration over time are needed. Acceleration sensors have to be used to detect component vibrations, while switches are needed to detect the amount of turning starts and shutdowns in a component’s lifecycle. The most interesting monitoring components of a plant, where resource consumption varies, are filters, actuators, areas of heat exchange and components, where electrolytic activities take place (e.g. semiconductors). These components

need to be particularly equipped with additional sensors. A list of important resources and sensors required for these measurements is presented in the following:

Natural Resource	**Sensor**
energy	smart meter
water	flow rate sensor
heat	temperature sensor
hydraulic fluid	flow rate sensor
lubricants	ultrasonic, temperature sensor and shock pulse
water vapor	valve position, temperature and pressure sensor
pressured air	valve position, temperature and pressure sensor
extracted air	valve position, pressure and flow rate sensor
coolant	temperature and flow rate sensor
time	timer
CO_2 emission	flow rate sensor
component waste	acceleration sensor, switch

Table 2.1.: Important resources of the manufacturing domain and sensors needed to measure them

Another monitoring aspect, which increases the complexity of RMSs, is that there is no unambiguous assignment of resources to components, which means that components can consume very different kinds of resources depending on their type. Therefore, it is not possible to make general assumptions about component types and their resource usage. Depending on the resources consumed by the component, the performance and characteristics of the component may vary. For instance, robots can have three possible kinds of drives:

- **Hydraulic Drives:** uses oil to produce energy; *advantage:* allows good reactivity and high accuracy; *drawback:* oil leakage or oil contamination are common problems
- **Electrical Drives:** electrical energy is transformed in mechanical energy; *advantage:* easy to control torque and speed; *drawback:* low force
- **Pneumatic Drives:** pressurized air is used to produce energy; *advantage:* very high reactivity, high force and low implementation costs, *drawback:* control system is complex, high energy consumption and bad CO2-footprint

In this case, an RMS has to be configured individually depending on the kind of drive. Thus, no general assumptions can be made such as the assumption “the energy efficiency of robots increases with torque and speed of its drive”. Since this rule holds only for robots with electrical drives. This means that type information about components is an essential part of the knowledge-base which has to be considered by the RMS.

2.1.2. Partial Plant Models

During the plant design process, various design decisions about the installation of plant components, the structure of the control system, etc. have to be made. These decisions involve several engineering disciplines, e.g. mechanical, electrical or software engineering. Each discipline uses their own engineering tool and standard of the heterogeneous industrial tool landscape with incompatible data formats to store knowledge about the plant. The discipline-specific knowledge is stored in **partial plant models**, e.g. a *structure model* describing structural facets about the containment hierarchy or a *process model* specifying sequences of activities executed by the plant. Such partial plant models are used by various applications in an industrial plant, e.g. control, monitoring or diagnosis systems, but currently, these models are not integrated in one single model that allows an entire view on all aspects of the plant.

Experts with different roles have role-specific demands on a configuration tool for defining partial plant models. These roles involve knowledge engineers, engineering experts, third party suppliers, component manufacturers, plant owners and operators. Current research usually focuses on defining partial plant models that correspond to the manufacturing goals from the viewpoint of one expert of a specific domain [160]. We focus on the joint consideration of the various perspectives of all domain experts since this enables us to develop manufacturing systems with broader functionalities.

2.1.3. Monitoring State

Each monitored plant element *c*, which is part of an industrial plant is residing in a monitoring state computed by monitoring rules at time *t*. The monitoring state can only change if the monitoring conditions change. The monitoring state should contain sufficient background information to allow for optimizing the resource efficiency of the plant element. The *monitoring state* used within this thesis is composed of two parts:

1. The *state tupel* $tup_c(t)$ contains all information of a plant element that is necessary to compute the monitoring state;
2. A *state category* cat_c describes a range of values for which the plant element has common behavior.

For a plant element c, *Cat(c)* is the set of all possible monitoring states. $cat_c(t) \subseteq Cat(c)$ is the state category of plant element c at time t.

For example, let motor m1 be a component with the monitored properties *rotational speed n*, *energy efficiency* η, *executed activity a*, *context ctx*.

Motor m1 can reside in three distinct state categories. The set of state categories for motor m1 is defined as:

- Cat_{m1} = { energyEfficiencyOk, WarningEnergyEfficiencyLow, ErrorEnergyEfficiencyLow }

The state tupel of m1 for a time t is a record:

- $tup_{m1}(t) = \{n = 200rpm, \eta = 0.8, a = runFast, ctx = energyEfficientProduction\}$

The state category of m1 for a time t is computed by a monitoring rule based on the information contained in the state tupel and is:

- $cat_{m1}(t) = \{\text{WarningEnergyEfficiencyLow }\} \subset Cat_{m1}$

The monitoring state of m1 for a time t is defined as a combination of the state tupel and the state category, such that:

- $state_{m1}(t) = (tup_{m1}(t), cat_{m1}(t))$

State function

A state function S is defined as the deterministic mathematical relation between a state category *cat* and the state tupel *tup*. S can be thought of as an operator that, applied to the input *tup* (e.g. executed activity, context, rotational speed, etc.) generates the category *cat* (e.g. WarningEnergyEfficiencyLow). This state function is shown in equation (2.1) for a plant element c (the mapping performed by the operator on the quantity enclosed within the brackets $\langle . \rangle$).

$$cat_c = S\langle tup_c \rangle \tag{2.1}$$

2.1.4. Monitoring Rule

Rules in general are used to perform reasoning on a given set of data to compute new facts. Within this work, rule-based inference systems are considered as used in symbolic Artificial Intelligence with declarative rules that follow a logic-based semantic. Monitoring rules are used to compute monitoring states based on monitoring conditions.

A rule consists of a THEN-clause and specifies the goal, head or consequent of the rule and an IF-clause, also named body or antecedent of a rule, consisting of the sub goals. A rule without a body is called a fact; a rule without head a query. All monitoring rules in a rule set are combined to derive answers to specific monitoring queries.

For the following examples a description logic syntax is used with variables denoted with "?". A basic example for a rule is

```
RULE basicRule:
  p(?a) ← q(?a), ¬(r(?a,?b))
```

The part p before the arrow is the THEN-clause. The part after the arrow is the IF-clause. Several rules can be combined to a rule base.

The following is a simple example for a monitoring rule base:

```
RULE Rule1:
  monitoringElement(?x) ← monitors(?a,?x), dataSource(?a)
```

```
RULE Rule2:
  monitors(?a,?x) ← monitors(?a,?v), part-of(?x,?v)
RULE Rule3:
  residesIn(powerTooHigh,controlUnit1) ← monitors(?sensor,controlUnit1),
  measures(?sensor, ?value), hasUnit(?value, W), ?value > 500
```

The first rule can be read as ?x is a monitoring element if ?a is a data source and ?a monitors ?x. The second rule defines the behavior of the monitors relationship - stating that ?a monitors ?x, if ?a monitors ?v and ?x is part of ?v. A third rule computes the monitoring state “powerTooHigh” for the controlUnit1. The unit of the measured value of the sensor is defined in "W" which is the symbol for the physical unit "Watt".

To demonstrate the computation of Rule 1 and Rule 2, the following facts are specified:

```
monitors(smartMeter1,controlUnit1).
part-of(motor1,controlUnit1).
dataSource(smartMeter1).
```

Based on this facts, Rule 2 would compute a new fact: `monitors(smartMeter1, motor1)` This new fact can then be processed by Rule 1. A query to this rule base could ask for all monitored elements in a plant such that:

```
monitoringElement(?a)
```

With the facts and rules above the evaluation of this query would return:

```
?a = motor1.
```

2.2. Industry Automation

Important concepts of the industry automation domain are shortly introduced in this cluster. These domain-specific concepts form the basis for constructing an RMS specific to the needs and constraints of the industry automation domain.

2.2.1. Functional Plant Hierarchy

Today, many systems correspond to the control hierarchy as defined in standard IEC 62264[1]. This standard defines a functional hierarchy model of a manufacturing enterprise as shown in Figure 2.2.

Different levels of the functional hierarchy model are depicted there: the *enterprise level* for business planning and logistics (level 4), the *process production level* for manufacturing operations and control (level 3), the *process and automation level* for batch, continuous, or discrete control (level 1&2), and the *field level* for the actual production process. The levels provide different functions and work in different time frames. The RMS defined within

[1] IEC 62264-3 (2007): *Enterprise-control system integration*

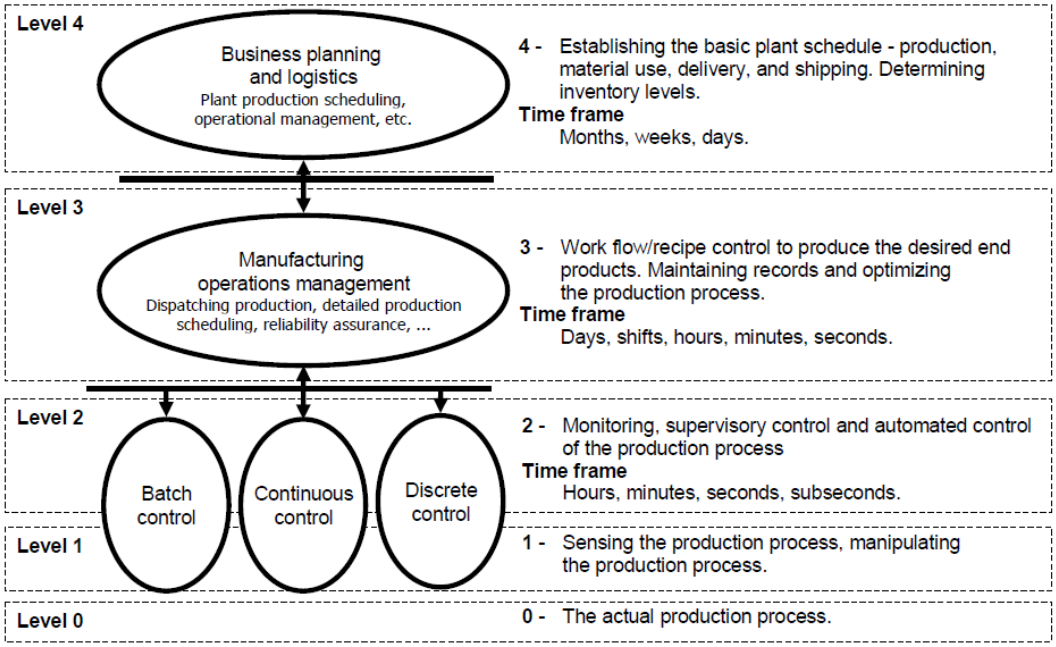


Figure 2.2.: Functional plant hierarchy (from Figure 2 in IEC 62264-3)

this thesis uses varying communication systems, tailored to meet specific requirements at individual levels of the functional plant hierarchy.

2.2.2. Roles of Experts in the Industrial Domain

Experts from different domains with local terminologies have to work together to develop and operate software-intensive industrial systems, like monitoring or control systems. Within this thesis, a differentiation between the following roles of experts is made:

- **Knowledge engineer**: specifies the partial plant models and describes or discusses specific concepts and terms with experts of different engineering disciplines (engineering experts).
- **Plant owner**: defines overall management goals, perspectives and constraints that need to be respected by all other stakeholders working at his plant.
- **Engineering expert**: is an expert in one or several engineering disciplines (e.g. electrical engineering, mechanical engineering, software engineering) that defines systems for different levels of the functional plant hierarchy together with other engineering experts.

- **Supplier**: specifies general information on abstract component types, e.g. product specifications as defined in product catalogs. Examples are manufacturers of components or third party suppliers.
- **Plant engineer**: mounts an industrial plant built on components of the supplier. During the plant engineering process, he specifies concrete plant knowledge and stores it in plant-specific models.
- **Plant operator**: operate an industrial plant and use information specified by plant engineers during plant engineering at operation time.

2.2.3. Industrial Automation Systems

The most important industrial automation systems related to this thesis are discussed in the subsequent sections: maintenance, condition monitoring, diagnostics and energy monitoring systems.

Maintenance System

Main industrial automation systems related to this work can be subsumed by the term *condition-based maintenance (CBM)*. Within this thesis, the definition of [22] is used, which says that CBM is "the monitoring of machines for the purpose of diagnostics and prognostics". The condition monitoring task here is mainly to recommend maintenance actions based on the collected information by comparison of the actual system behavior with the behavior predicted by a model [60]. Diagnosis can be divided into two parts: fault-detection and fault-diagnosis [32], while fault-detection is equivalent to monitoring in the context of this thesis. Symptoms of possible failure can be detected early or, once a fault happens, the fault diagnosis can be improved by using the monitoring results. Both, diagnostics and prognostics systems are two aspects of CBM. All CBM aspects have in common that they consist of three key steps (see Figure 2.3) according to [65].

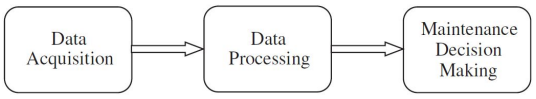


Figure 2.3.: Three steps in a CBM program specified by [65]

Condition Monitoring System

Condition monitoring of components in the plant environment is gaining importance in industry due to the need to increase the reliability of components and to decrease the possible loss of production due to component breakdown. According to a definition in

[149], condition monitoring is “necessary to access the state of a machine and to determine whether it is **malfunctioning** through reason and observation”.

Interviews with experts of the industrial domain conducted in the context of this thesis showed that resource monitoring is often compared to condition monitoring. Some domain experts suggested to use research on CMS as a basis to this work. To a certain extent this advice was helpful, however in detailed discussions and interviews with experts, the author of this work identified multiple differences between these two monitoring types which impedes simple reuse of research results on CMS. All identified differences are discussed in the following.

First of all, the goal of condition monitoring systems is to ensure a continuous and reliable production to maximize the component life, whereas the goal of RMSs is to optimize the usage of resources. A RMS requires, thus, not only measurement of sensor data for threshold comparison, but also an awareness of the components' environment and situation. Second, RM requires additional sensors that are currently not installed in industrial plants, e.g. a power meter to measure the reactive, active and apparent power consumed by every component or a manometer to measure pressurized air produced by air compressors. The task of CM is to identify the state of single components and warn the operator if an erroneous state is detected, so that he can react immediately. States of single components are usually not critical for RMSs, since a plant can still operate when the resource consumption exceeds a certain limit. However, the operator might be overcharged if an RMS reports the state of every single components constantly, he thus needs to get an overview over the entire plant by means of state aggregation. The computation of states for composite components or the entire plant is thus required for resource monitoring. Another difference arising when monitoring resources is that resource monitoring parameters (e.g. power or pressure) may oscillate at high time frequencies and are not as stable as parameters usually used for condition monitoring (e.g. temperature or vibration). The detection of power peaks, for example, requires a minimum sampling frequency of 10kHz as shown in [67].

Diagnostics System

In contrast to condition monitoring, the purpose of diagnostics is to determine **causes** of faults (instead of determine malfunctioning) by predefined diagnosis rules. More specifically according to [65]: “Diagnostics deals with fault detection, isolation and identification, when it occurs.”

In spite of what may appear at first glance, monitoring and diagnostics systems are quite similar. Various research approaches [154, 50, 47] suggest to build frameworks that can be used for both monitoring and diagnostics. This is due to the identical key steps needed to realize both industrial automation systems as shown in Figure 2.3. An example are rule-based monitoring or diagnostics systems; monitoring systems use sensor and process data as rule input and the rules determine monitoring states, while in diagnostics scenarios the input are faulty monitoring states and the rules determine causes of faults.

The fundamental concepts and ideas developed within this thesis can thus also be applied for diagnosing faulty behavior of the plant's resource usage.

Energy Monitoring System

Energy monitoring shall ensure that the key characteristics that determine energy performance are monitored, measured and analyzed at planned intervals as defined by ISO 50001[2]. According to [59, 96], typical functionalities of energy monitoring systems are: 1) measurement and preprocessing of energy data (active/reactive power, tension, electricity, etc.), 2) permanent storage of energy data, 3) visualization and analysis of energy data considering multiple criteria (duration, time, frequency distribution of measured properties), 4) prediction and planning of future energy demand.

In a broader context, energy monitoring systems are one aspect of *energy management* strategies of companies. The following standards define requirements for implementing energy management systems in companies:

- ISO 50001: Defines organizational requirements for implementing energy management systems: energy policy, establish objectives, and action plans
- VDI 4602 [3]: Defines crucial terms, gives examples for actions in energy efficiency
- VDMA 66412[4]: Performance evaluation by MES KPIs with regard to energy efficiency

Energy monitoring is a subtype of resource monitoring and can be covered by means of the approach defined within this thesis. This becomes evident when considering application scenarios AS1, AS3 and AS5 presented in Chapter 15. These resource monitoring scenarios address also the functionalities required by energy monitoring systems.

2.3. Representation Languages and Paradigms

This cluster defines important knowledge representation languages and paradigms affecting the technological basis of this work.

2.3.1. Process Specification Language (PSL)

The Process Specification Language (PSL) has been designed by [57] to facilitate correct and complete exchange of process information. It was developed by the Technical committee ISO TC184 for Industrial automation systems and integration, and published as ISO

[2] ISO 50001 (2011): *Energy management systems – Requirements with guidance for use*
[3] VDI 4602-1 (2007), 4602-2 (2011): *Energiemanagement - Definition, Begriffe*
[4] VDM-Einheitsblatt 66412 (2011)

standard in the document ISO 18629[5]. It builds on first order logic and is designed in a modular way, i.e. its axioms are organized into PSL Core and a set of extensions.

The following extensions are used by the process model specified in this work and discussed in more detail in Section 8.3:

- *PSL Core:* defines basic concepts of PSL
- *Subactivity Extension:* describes how to aggregate and decompose activities
- *Activity-Occurrences Extension:* defines relations that allow the description of how activity-occurrences relate to one another

2.3.2. AutomationML

AutomationML (Automation Markup Language) is a data format based on XML for the storage and exchange of plant engineering information [9]. It is provided as open standard and specified in more detail in [41]. Its goal is to interconnect the heterogeneous tool landscape of modern engineering tools for different engineering disciplines, e.g. mechanical plant engineering, electrical design, HMI development, PLC, robot control, etc. AutomationML is based on other data formats such as CAEX, COLLADA and PLCOpen.

The standard CAEX (Computer Aided Engineering Exchange) is a recognized standard data exchange format for plant engineering data according to [116]. IEC 62424[6] specifies CAEX as being a meta model for the storage and exchange of engineering models. Below, its addressed topics are briefly summarized.

- Description of a concrete containment hierarchy of components (*InstanceHierarchy*), from top-level plant down to single components (*InternalElements*) with interfaces (*ExternalInterfaces*) and relations (*InternalLinks*).
- Reusable *SystemUnitClasses* defining component types down to their respective technical realizations organized in vendor-specific product catalogs. Therein, hardware components are detailed by the vendor.
- Reusable role definitions for abstract descriptions of components (*RoleClasses*).
- Reusable *InterfaceClasses* for specifying connection points of *RoleClasses*, *SystemUnitClasses* and the interface type of *ExternalInterfaces*.
- *Attributes* for describing characteristics of each previously introduced modeling element.

[5] ISO 18629 (2004): *Industrial automation systems and integration -Process specification language*

[6] IEC 62424(2008): *Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*

2.3.3. Meta Object Facility (MOF)

The Meta-Object Facility is an Object Management Group (OMG) standard[7]. This standard is implemented on a highly extensible Model Driven Architecture (MDA) framework and very successful in software engineering applications. It is based on four modeling layers for defining different levels of abstraction. Such a Model-Driven Architecture for the representation of layers allows engineers to participate in the modeling process in a collaborative manner and to ensure a formal representation of the expert knowledge as shown in [75].

The key modeling concepts of MOF are ***type*** and ***instance***, and the ability to navigate from an instance to its metaobject (its type). The conceptual modeling approach developed within this thesis is grounded in MOF, in the sense that it is defined in terms of the MOF meta-metamodel.

2.3.4. Unified Modeling Language (UML) and Systems Modeling Language (SysML)

The Unified Modeling Language (UML) driven by the OMG is a fundamental component of MDA framework. The difference to the MOF is that UML is based on three modeling layers instead of four. UML is the unification of many existing object-oriented graphical modeling languages. By now, UML has become a well-established and popular standard aiming at designing and describing system and domain models. UML is mainly applied in the following chapters for a visual syntax for models and model mappings.

The Systems Modeling Language (SysML) is a profile of the UML, which is smaller and better suited for model-based systems engineering and thus widespread and more common in industry. A drawback of SysML is that the diagrams are not well-suited for presenting the overall view and the different hierarchy levels of an industrial plant as found in [55].

2.3.5. Web Ontology Language (OWL)

The Web Ontology Language (OWL) [105] has been standardized by the W3C consortium in February 2004 as a language for modeling and semantic annotation of web content. Since, it has become the most-accepted ontology language in the Semantic Web community and is supported by a constantly increasing number and range of tools and applications, while it is also designed as an extension to the Resource Description Framework (RDF) [69].

There are three official sub languages of OWL, which are *OWL Full*, *OWL DL* and *OWL Lite*. The most wide-spread sub language is OWL DL (DL for "description logic"), since it is computational complete (all conclusions are guaranteed to be computed) and decidable (all computations will finish in finite time). OWL DL is fully supported by most software

[7] ISO/IEC 19502 (2005): *Information technology - Meta Object Facility (MOF) for model-driven engineering*

tools and therefore used within this thesis for describing plant models. Currently, OWL 2 is available as updated version of classical OWL with additional features specified in [100].

2.3.6. Concept Reification

In general terms, reification allows the "formation of descriptions of descriptions" as stated in [24]. In turn, this allows descriptions not only of relationships, but also of abstractions about them, i.e. classes of relationships expressed without regard to any particular explicit relationship. RDF and OWL allow such reification, e.g. by providing a built-in vocabulary intended for describing RDF statements [49] (Statements may be subjected to reification by assigning a URI and usage of `rdf:Statement`, `rdf:subject`, `rdf:predicate`, `rdf:object`).

A specific type of reification is *concept reification*. Concept reification means that a given class can be regarded as the instance of a higher level meta-concept (or class) or that an object can be a class and an instance at the same time as defined in [11]. Concept reification is necessary whenever a high degree of reusability has to be guaranteed in a knowledge-based system or if a model needs several metaconcepts as defined in the MOF standard. The conceptual modeling approach developed within this thesis is grounded in MOF, which means that concept reification is supported.

3. State of the Art

Research on resource monitoring in the manufacturing domain using knowledge-based approaches is still a “white spot” in the research landscape. An identification of state of the art approaches from related research fields regarding the described problem was thus conducted. When identifying related research fields, it became obvious that several distinct research fields touch parts of the problem. The research fields relevant to this thesis were thus divided in three categories presented in Figure 3.1. Research on business scenarios such as monitoring in the industry automation domain focuses mainly on knowledge-based, data-driven or analytical approaches. Besides, approaches related to monitoring can be divided into condition, energy and resource monitoring approaches. Section 3.1 proposes research on monitoring approaches in the industry automation domain, various knowledge-based solutions in the manufacturing area are presented in Section 3.2 and energy management research results are discussed in Section 3.3.

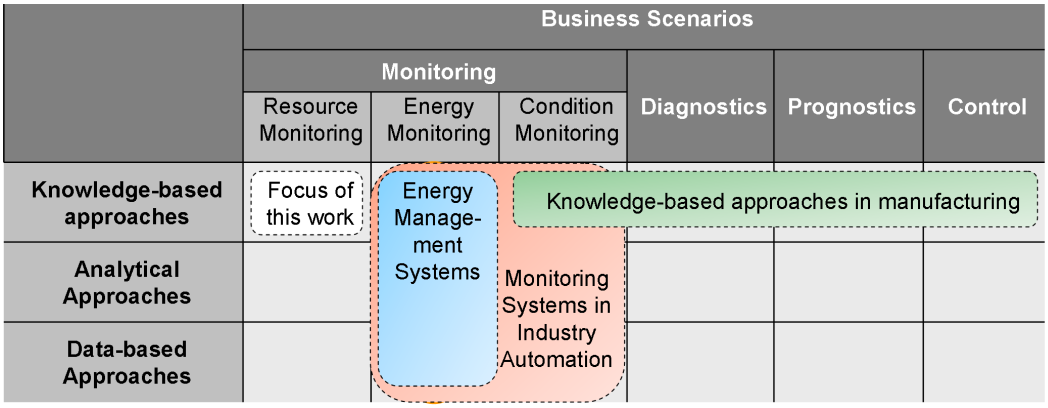


Figure 3.1.: Research fields related to this thesis

3.1. Monitoring Systems in Industry Automation

CHIANG ET AL. point out that monitoring systems in industry automation proposed in literature are usually implemented based on *data-driven, analytical*, or *knowledge-based modeling* techniques [30]. These techniques are introduced in the following by giving prac-

tical application examples. The advantages and drawbacks of these approaches identified in [54] are then discussed with respect to resource monitoring.

The main concepts and approaches of *data-driven modeling* are based on computational intelligence and machine-learning methods such as neural networks, support vector machines (SVM) or genetic algorithms [132]. Examples for data-driven monitoring systems using SVM were summarized and reviewed in [157] and two data driven techniques, neural network (NN) and principal component analysis (PCA), were integrated in [29] for process monitoring.

Analytical approaches generally involve detailed mathematical models, e.g. Hidden Markov Models, used for parameter estimation or observer-based methods. For instance, a condition monitoring approach based on Hidden Markov Models was specified in [52]. Additionally, [155] present a case study where parameters were estimated based on sensor observations from condition monitoring. This case study showed that such parameter estimation techniques can be properly established for condition monitoring systems.

Technologies associated with *knowledge-based approaches* appropriate for monitoring include expert systems, fuzzy logic and rule-based systems. For example, [42] successfully adapted an expert system for vibration analysis in machine condition monitoring. Different types of fuzzy logics for monitoring and diagnostics of industrial plants were evaluated and compared in [28].

An evaluation of these three approaches in regard to condition monitoring in [54] revealed that no single approach is satisfactory in every respect. The results of this evaluation are presented in Figure 3.2. While model-based techniques are an example for knowledge-based approaches, feature extraction techniques are based on analytical approaches and neural networks are an example for an implementation of a data-driven approach. Ideally, these approaches should be integrated in one unique system as suggested in [1]. First attempts to combine these approaches in process control industries are reviewed in [145]. To combine them in an effective architecture of the RMS, the suitability of every method for resource monitoring is evaluated.

In contrast to the other techniques, model-based approaches are said to be applicable efficiently for newly developed systems. But currently, there is no measured data available for resource monitoring since this is a new application field and the plants are often lacking required sensors, e.g. smart meters for measuring energy consumption. Data-driven and analytical approaches strongly depend on the availability of initial data and are, therefore, only suited for application to existing machinery.

An important advantage of model-based approaches compared to the other approaches is their high adaptation and maintenance capability. This capability is important to automatically adapt RMS in order to react on environment changes in flexible production systems. Such adaptations or maintenance tasks involve much manual work and are thus very costly in terms of time and personnel. As pointed out in [78], these costs can be reduced by means of model-based approaches.

Another problem of data-driven and analytical approaches is that they are language-dependent. But as stated in [135] a huge variety of different specification techniques

	Model based	Feature extraction	Neural networks
1. Required process knowledge	extensive	limited	minimal
2. Required initial data	minimal	limited	extensive
3. Development effort	extensive	extensive	limited
4. Computer power during development	limited	limited	extensive
5. Computer power on-line	extensive	limited	minimal
6. Maintainability	good	moderate	bad
7. Adaptability	good	moderate	bad
8. Acquired in-sight	extensive	limited	none

Figure 3.2.: Comparison of condition monitoring methods [54]

and languages are used within the heterogeneous environment of industrial automation. An RMS designed for the industrial automation environment should thus be language-independent and support mappings of terms from different languages.

3.1.1. Summary

Due to the reasons summarized in the previous section, knowledge-based approaches are the most appropriate approach for resource monitoring systems in industrial plant. The only issues experienced with knowledge-based approaches are a high development effort and the need of extensive process knowledge. A systematic engineering approach is thus needed to decrease the effort for the plant engineering experts to design a knowledge-based RMS and insert the required process knowledge. Usability aspects, such as ease-of-use and learnability of the knowledge-based RMS, are an important validation criteria. The work presented in this thesis is intended to cover the commented issues.

3.2. Knowledge-based Approaches in Manufacturing

The usage of knowledge-based approaches for engineering and maintenance of automation systems in the manufacturing area has been a major trend over the last years. While there is a considerable amount of research on applying knowledge-based approaches to individual problems in manufacturing such as condition monitoring of electro hydraulic linear drives [134] or gas turbines [90], the dependencies between those solutions and definition of application-independent solutions is less well investigated. Examples of knowledge-based modeling approaches in the manufacturing domain are given in the following.

SCHLEIPEN ET AL. present an approach for automatic configuration of a production monitoring and control application. More specifically, a concept for the automatic configu-

ration of the production monitoring and control system ‚ProVis.Agent' in the manufacturing domain based on CAEX is discussed [116]. One major drawback of CAEX-based approaches is that XML follows a hierarchical model structure, which is more restricted than knowledge-based approaches based on Semantic Web paradigms. So, to improve performances in discovering and validating information, i.e. to implement a consistent RMS, there is a need to add value in the field of relationships between objects. First approaches to overcome these drawbacks have been investigated, e.g. in [112].

CHRISTIANSEN ET AL. present a plant-wide diagnosis systems using process knowledge in addition to structural knowledge of the plant [32]. Their diagnosis systems aims at identifying root causes of faults. The diagnosis system is tested in manufacturing and process industry applications. They showed that usage of process-specific information for diagnosis is beneficial since it yields to an enlargement of the amount of possible root causes. A related approach using practical tools for grouping alarms in an industrial process control system is defined by [114]. The focus of this approach is compliance with standards for alarm management which set limits on the number of alarms per unit time for an operator. These applications demonstrated advantages in combining knowledge about different plant aspects for industrial automation systems.

LEGAT ET AL. propose a knowledge-based system for monitoring and control of industrial plants. This approach encompasses a formal logic-based model for flexible information acquisition from a Plant Lifecycle Management Systems (PLMS) and an automated reasoning mechanism [79]. An example of how the proposed approach can be realized for the diagnosis of specific automation components using the Siemens PLMS product COMOS is further presented. A similar rule-based approach for flexible control systems in manufacturing using Autonomous Product Memories is defined in [121]. The task of the rules here is to specify the content of the product memory in a decentralized system architecture. This approach gives hints for developing a decentralized, rule-based architecture as basis for implementing a knowledge-based RMS.

LOSKYLL ET AL. describe a semantic service discovery and orchestration system to provide a concept towards the creation of adaptive production processes [83]. The main idea of this approach is to encapsulate functionalities of components of a production plant in web services, which can be invoked via standardized communication interfaces. The approach is based on different semantic service technologies which are experienced during practical implementation. [82] suggests the usage of contextual information (e.g. the status of the plant or its components, special sensor readings) for automation systems to infer reasonable configurations and find appropriate combinations of basic functions to realize a desired functionality for a specific application.

VIINIKKALA ET AL. apply Semantic Web ontologies for a maintenance demand analyzer. The task of this analyzer is to produce a maintenance recommendation for devices in specific processes based on web services [147]. To provide ease of use, a generic and a specific query interface were implemented by the plant model services. These queries are used here instead of rules to compute maintenance recommendations. In contrast

to queries, rules create new facts in the knowledge-base which is required by intelligent resource monitoring applications including historical data analysis.

PAKONEN ET AL. study the use of Semantic Web and information agent technologies in the context of industrial process monitoring [102]. Their monitoring system aims to increase the overall situation awareness by combining information from various heterogeneous data sources using a common ontology. Their findings point out that for determining the operational situation of a monitored industrial process, an operator needs access to a wide range of information stored in various IT systems such as electronic diaries or maintenance databases. In [101], the information retrieval of the monitoring system was enhanced by using fuzzy ontologies to properly address uncertainties, inconsistencies or contradictions. In contrast to the RMS defined within this thesis, the industrial process monitoring applications proposed here are limited to the MES layer of the automation pyramid.

BERNARD ET AL. implement a rule-based system for process control on the MIT research reactor [14]. Result of an evaluation showed that rule-based systems are generally more robust than their analytical techniques. Their findings suggest to complement analytical techniques with rule-based systems to improve the system performance.

CALDER ET AL. define a semantic data validation tool capable of observing incoming real-time sensor data and performing reasoning against a set of rules [24]. These mechanisms validate sensor observations to test hypotheses about anomalous sensor network observations in coastal ecosystems. This reasoning approach seems to have a high performance for processing of real-time sensor data. However, they did not yet examine the scalability of the system in a real time environment.

DAI ET AL. propose an approach to semantic analysis of control systems using multiple-layered ontological knowledge representation and a rule-based inference engine [36]. The ontological knowledge base is automatically generated from the IEC 61499 standard. Pointing out that usage of description logic (DL) guarantees decidability of the task of classification, this work showed that DL reasoning is scalable for reasonably large realistic plants.

3.2.1. Summary

Currently, there is a big variety of engineering tools in the manufacturing engineering chain with proprietary or incompatible data exchange formats [119]. For this reason, efforts were made to introduce new data exchange formats such as the system-independent format CAEX for the engineering and maintenance of production systems [111]. However, there are only few approaches that use either structural, context or process knowledge or even a combination of them for the task of monitoring an industrial plant. Most of the approaches are not application-independent, but designed for applications in specific branches of industry or manufacturing domains.

As stated in the introduction, context information is needed for intelligent resource monitoring. In the field of context awareness in manufacturing, only few research results were

published up to now. Most of the work in the field of context awareness is concerned with providing either a framework to support the abstraction of context information from the field level of the plant or high-level models of context information to provide context services [12]. These two levels need to be combined to provide more accurate context information for resource monitoring.

An important barrier for implementing knowledge-based, or more specifically rule-based systems, is the scalability and the system performance related to the reasoning task as argued in [16] and [13] respectively. Several knowledge-based approaches presented in the last section found ways to overcome this barrier. These approaches give hints how to implement a scalable, high performing knowledge-based RMS.

3.3. Energy Management Systems

Monitoring the energy demand and energy efficiency in manufacturing are intensively discussed aspects in research and industry. Questions regarding energy efficiency and energy management are interpreted differently in diverse research domains and on different abstraction levels in factory automation. Since decisions concerning energy-related issues need to be made within this thesis, related state-of-the-art approaches were selected and discussed in this section.

HU ET AL. propose an efficient approach for energy efficiency monitoring of machine tools based on mathematical models [61]. The energy efficiency monitoring model of this approach is constructed based on an energy consumption model of machine tools divided into two parts, i.e. constant energy consumption and variable energy consumption. diameter). An application example of the system on a CNC lathe is given, but the approach can be extended easily to other kinds of machine tools. An important limitation of this method is the requirement of several experiments for the definition of the energy consumption model.

VIJAYARAGHAVAN ET AL. investigate machine tools with regard to energy monitoring. The authors of this paper point out that manufacturing processes are strongly related to the energy demand of machines in complex manufacturing settings. The objective is to correlate energy usage with operation of the manufacturing system. Therefore, event stream reasoning techniques are applied in order to monitor energy input patterns in large systems [148]. The approach enables the concurrent monitoring of energy demand related to process data on different levels of analysis. Reasoning is based on a rule engine respectively complex event processing engine implementing the Rete algorithm. Different modes with related energy demand of a machine (start up and shut down, idling, and processing modes) can be distinguished in this way. In contrast to this thesis, they focus on event stream processing techniques for monitoring.

DEVOLDERE ET AL. discuss energy aspects for two discrete part producing machines types and proposed an initial design improvements to reduce the overall energy consumption [38]. This type of analysis is the starting point in designing machines that have fixed energy loads and per-part energy loads. Although the studies do not reveal the absolute

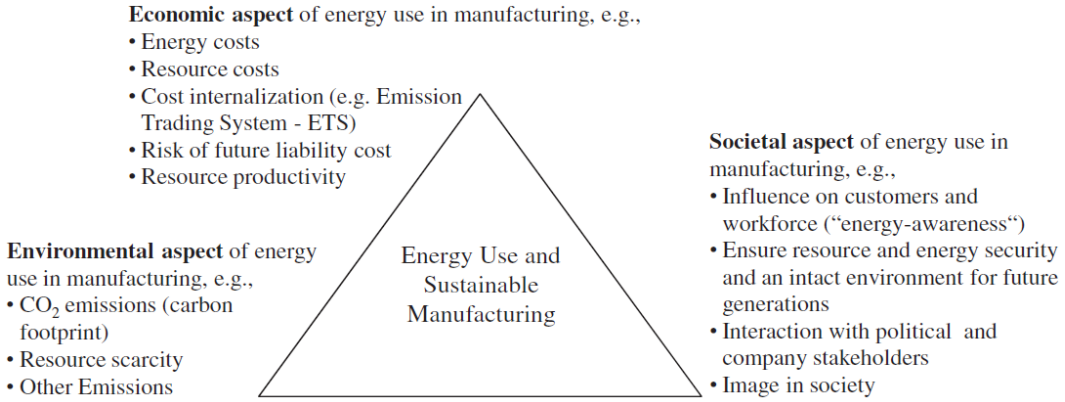


Figure 3.3.: Contribution of energy management systems to main aspects of sustainable manufacturing according to [23]

time intervals of the machines, the studies give indication to energy demands of industrial machine tools.

BUNSE ET AL. point out that energy management contributes to three main aspects of sustainable manufacturing as presented in Figure 3.3. However, there exists a gap between the energy management solutions available and the actual implementation in industrial companies [23]. Measurement of energy efficiency of industrial plants is the first step for evaluating and implementing energy improvement measures in industrial plants. Furthermore, energy efficiency manufacturing metrics are needed to identify inefficiencies within a plant's energy usage (e.g. energy consumption profiles). Other findings of the study were that processes are required to map energy usage for better understanding input, output, and measurement points for each manufacturing process and the definition of benchmarks for similar components should be facilitated. Since energy management systems are strongly related to resource monitoring, these requirements need to be addressed by RMSs.

SIVILL ET AL. state that many business organizations have started to control and manage their energy performance on a continual basis, but one of their major questions is how to prioritize results of energy performance measurements? To find an answer to this question, they carried out interviews with managers and operators in three energy-intensive industrial sectors in Finland [131]. As a result, they identified important research and development needs of energy performance measurement that are required to further improve energy performance in industry. Evaluation of a questionnaire on the utilization purposes of energy-related information with different stakeholders showed that important energy management factors are to "identify opportunities for structural changes and process integration" and to "monitor energy consumption compared with its expectation values".

3.3.1. Summary

The amount of research on energy management systems illustrates the importance of considering energetic aspects of industrial plants. State of the art approaches have been reviewed with regard to energy planning, monitoring, and control. These approaches showed that one major success factor of these systems is to systematically identify important energy management factors, such as utilization purposes of energy-related information first, to respond to the demands of industrial practice. Transferring these findings on resource monitoring, this means that resource monitoring factors need to be identified to guide the design of an efficient RMS.

Part II.

Resource Management Factors

4. Empirical Basis

A state of the art review on energy management approaches presented in Chapter 3 showed that a major success factor of energy management systems is to systematically identify important Energy Management Factors to capture challenges that need to be tackled in order to respond to the demands of industrial practice. Transferring these findings on RMSs, this means that important **Resource Management Factors (RMFs)** need to be identified prior to establishing a knowledge-based RMS appropriate for addressing the challenges of the manufacturing domain.

This chapter introduces the empirical basis for the analysis of RMFs that knowledge-based RMSs for industrial plants have to face; to generate hypotheses for its explanation and to derive a knowledge engineering methodology and requirements for the RMS that can help tackle these factors. The results of empirical studies conducted at the engineering and electronics company Siemens AG are provided in the following chapter. These studies were conducted to identify practical RMFs based on engineering challenges of the industrial domain. By means of the identified RMFs, a knowledge engineering methodology and system requirements were derived. This is essential to ensure that the solution approach to be developed in this thesis is applicable to the demands of industrial practice.

The empirical basis consists of the results derived in empirical studies with different sources of evidence. These sources of evidence were continuously used during this thesis on different levels depending on the degree of knowledge already available. The author was involved in multiple meetings within the RES-COM project where he observed and interviewed the participants of the meetings. Further opportunities for empirical studies by interviews and observations emerged by the author's supervision of student theses, and interviews with researchers at the Technische Universität München. According to the classification introduced by [146], these sources of evidence were the:

- conduction of interviews with experts of the industrial domain
- collection and interpretation of documents such as technical reports, standards and research publications
- direct and participant observation of production processes
- collection and analysis of physical artifacts (such as computer software and hardware)

In the following, RMF1-RMF8 identified by means of empirical studies are reported. The goal of these studies were to identify the issues the developers of knowledge-based

systems in the industrial domain; and more specifically the issues the developers of knowledge-based RMSs have to face. The study focused on the engineering phase that in the author's experience is particularly problematic for the definition of knowledge-based systems. These user-specific issues were generalized to formulate RMFs specific to the industrial domain and specific to monitoring. The industrial domain specific RMFs are more general and need to be tackled also by other kinds of manufacturing systems such as diagnostics or prognostics systems.

4.1. Resource Management Factors Specific to the Industrial Domain

Comparing the RMFs for the knowledge engineer in the industrial domain to other domains, similarities as well as differences could be identified.

RMF1 (Knowledge Acquisition Bottleneck): A challenge in all domains is to acquire the knowledge of the domain experts and the knowledge distributed in different sources in various formats (e.g. documents, diagrams, etc.) to form an appropriate knowledge base. This is a well known challenge in the area of knowledge engineering also known as the "knowledge acquisition bottleneck" [21]. During a cooperation project with the mechanical engineering department of the Technische Universität München, it turned out that difficulties arose when trying to access knowledge about industrial plants. The task within this project was to capture diagnostic knowledge about an industrial facility for fiber board production. A major challenge was to get information about the alarm structure of the plant from alarm systems currently in use and from interviews with the plant operators. The major problem was that their alarm messages were specified in multiple languages in different countries by distinct operators. The plant operators were only in charge of some specific plant aspects, e.g. the control structure of the plant. Thus, the most reliable knowledge source covering all alarm-related aspects were documents about the plant instead of operator knowledge. This participant observation showed that a major challenge for RMS is to capture and maintain huge volumes of knowledge which requires new ways of knowledge acquisition.

RMF2 (Shared Common Models): A challenge specific to the industrial domain is the integration of knowledge about an industrial plant in a shared common model. The knowledge is currently distributed among experts from several disciplines. Disciplines like mechanical engineering, electrical engineering, automation engineering and computer science have developed different ways of modeling as shown in [92]. Huge plant modeling tool vendors such as Siemens offer tools that usually focus on specific engineering disciplines, e.g. Siemens COMOS for integrated plant asset management or Siemens PLM NX for product development. Also vendor-independent tools are often specific to a certain

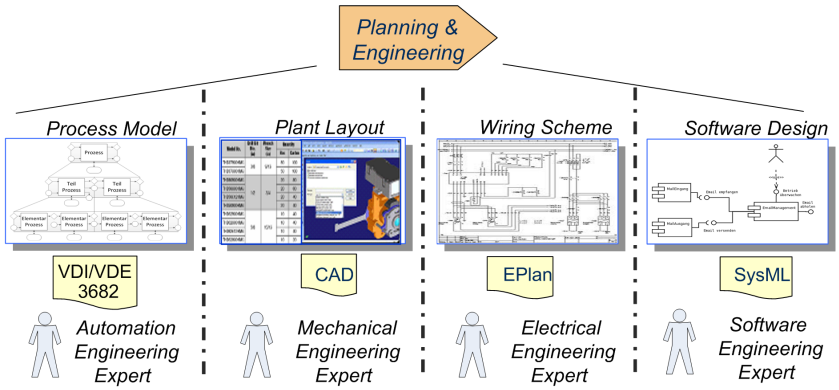


Figure 4.1.: Planning and engineering phase require different kinds of knowledge

discipline as shown in Figure 4.1, e.g. CAD tools are used for representing the structure of industrial plants or SysML is used for modeling the software design of industrial plants. As a consequence, there is not a shared common model for an industrial plant, but several models where each expert covers the scope of his expertise. In other domains, such as the medical domain, knowledge engineering is used to a higher degree, which may be a result of more shared common knowledge as stated in [156].

RMF3 (Standard Vocabulary): Not only the ways of modeling industrial plants are discipline-specific, but also the vocabulary and meanings of terms. This may be depending on the industrial area (e.g. discrete manufacturing vs. process manufacturing) or the knowledge sources (tools, standards, literature). Within the RES-COM consortium, problems arose when trying to identify basic terms for modeling concepts in the manufacturing domain. For example for the basic term *device*, we collected a list of four synonyms in a group of 10 people: *component*, *asset*, *system unit* or *resource*, while some of them have additional meanings (*components* may be parts of the plant or of the product, *resources* may also be materials like steel or supplies like energy). Furthermore, interviews were conducted with experts from the industrial domain to identify basic concepts to model structural plant aspects. Depending on the background of the interviewee, they used different terms and associated different meanings with these terms. Particularly the vocabulary of software engineers, electrical or mechanical engineers differed significantly. Another indicator which confirms our experiences made during the interviews and workshops is the huge amount of standards available for the industrial domain, e.g. MESA[1], VDI 5600[2], IEC 62424[3], etc. They define different terms that can be mapped on each others as

[1] MESA (2004): “MESA White Paper 08: MESA’s Next Generation Collaborative”

[2] VDI 5600 (2007): “Fertigungsmanagementsysteme (Manufacturing Execution Systems - MES)”, available at www.vdi.de/5600

[3] IEC 62424 (2008): “Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools”

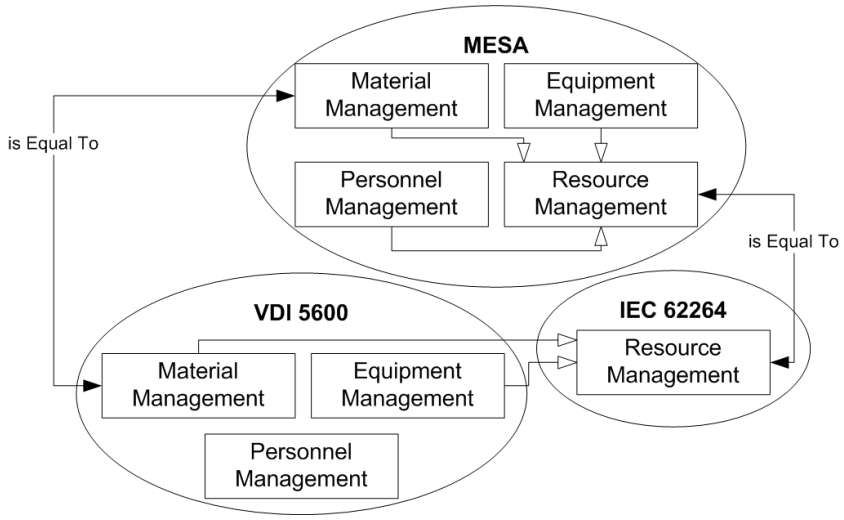


Figure 4.2.: Mappings of terms of different industrial standards

shown in Figure 4.2. Thus, models are needed which can handle synonymous terms and homonym terms (terms with several meanings), so that we can define a basic standard vocabulary for defining a domain model for the industrial domain that relates all terms by expressing their semantics. This is different from the medical domain, where a standard vocabulary is already used to describe the domain knowledge and the medical terms, such as diseases and symptoms.

RMF4 (Generic Plant Description): An additional RMF arises from the fact that every facility has a different structure and behavior, even if they manufacture the same product. This is different to other areas, such as the medical domain, where entities have similar structure and behavior for every situation (e.g. “all humans have arms attached to the shoulders”). Major differences between plant structures were experienced when comparing plants in distinct manufacturing areas. Examples of plants built within the RES-COM project (discrete manufacturing area) were compared to plants used within the I2MSteel project (steel manufacturing area). This comparison revealed that in the discrete manufacturing area, the components installed in the plant have one specific manufacturer, whereas the components installed in a steel manufacturing plant are often self-made or composed of smaller components of different manufacturers. Multiple other differences between plant structures in these research projects were identified which rises the need for a generic plant description that allows to cover all kinds of plants. This general description can then be individually adapted to describe specific details of concrete facilities.

RMF5 (High Degree of Reconfigurability): The manufacturing area is characterized by a changing environment which encompasses decreasing model life cycles, versatile pro-

duction (i.e. numerous product variants), and permanent change requests. Various studies concerning future manufacturing systems have shown that the manufacturing environment needs to become more flexible, e.g. [34] state that the era of mass production is currently being replaced by the era of market niches which makes reconfigurability of manufacturing plants increasingly important. Experts of the domain confirmed in interviews that modern manufacturing systems, such as monitoring, diagnosis or prognosis systems, need to allow a higher degree of reconfigurability to stay competitive within the market. Generally, knowledge-based technologies are said to offer a high degree of reconfigurability, but this is only the case if they are constructed in an extensible and flexible manner. The characteristics extensibility and flexibility are thus an important prerequisite for establishing a reconfigurable RMS.

4.2. Resource Management Factors Specific to Monitoring

Crucial RMFs specific to resource monitoring are listed in the subsequent section.

RMF6 (Sampling Rates): Modern monitoring systems, and especially RMSs, require a high sampling rate to ensure a continuous observation of all sensor and process data registered in a plant. During system tests, it was found that sampling rates of 10 Hz or higher are necessary to detect electricity peaks of engines. Electricity is a highly oscillating parameter and requires thus higher sampling rates than other monitoring data, such as temperature. These findings were confirmed by researchers of the Technische Universität München at the department of Energy Economy and Application Technology [144]. They pointed out that energy peaks are usually occurring during switching operations of devices and need to be detected by a monitoring system with exceptionally high sampling rates.

RMF7 (Implementation Costs): According to a report of the German government, the aim of their sustainability program is to increase the resource-efficiency of industrial plants which requires high capital investment [94]. Furthermore, an efficient usage of resources is necessary to guarantee the competitiveness of companies and the availability of natural resources on a long-term perspective. The implementation of resource monitoring systems in industrial plants is particularly expensive, because it requires time and manual effort. Additionally, the financial savings for the companies are currently small. This is a major obstacle for companies to investigate in systems in the context of resource-efficiency. This issue can only be tackled by defining an RMS, which guarantees a high level of reusability in order to keep the costs for adaptations and implementation low.

RMF8 (Complexity of Algorithms): The author of this thesis conducted interviews with engineering experts to identify wide-spread tools in the manufacturing domain. Then, most

common tools were reviewed in order to analyze the user behavior of engineers. Widespread tools for modeling or simulation in the industrial environment are for instance Matlab, Octave or AutomationML. It was found that engineering experts are usually describing characteristics of devices and facilities by mathematical expressions and the dynamic behavior of the plant by time series events. A monitoring system should allow engineering experts to define complex algorithms and time series analysis rules in such a manner, e.g. a rule that states “if the energy efficiency η of motor m_1 is lower than 60% for more than 1 minute, then the state of the motor m_1 is set to *error*”.

5. Evaluation of Empirical Results

The previous chapter outlined empirical results relating to which RMFs knowledge-based RMSs are facing in the manufacturing environment. It focused thus on answering the first research question:

> **Research Question 1:** What factors influence whether knowledge-based resource monitoring systems in manufacturing companies can be established and how can these factors be addressed by means of an appropriate methodology?

Parts of this question were answered in the previous chapter. In order to evaluate the identified resource management factors regarding their generalisability, they need to be compared to similar existing case studies. Thus, the following evaluation question was defined to validate RQ1:

> **Evaluation Question 1:** Can the identified resource management factors crucial for the successful establishment of knowledge-based RMSs in the manufacturing domain be affirmed by similar case studies?

The insights gained in the previous chapter regarding RMF1 coincide with the findings of further studies, identifying the acquisition of knowledge as main factor of success when implementing knowledge-based systems, causing important knowledge to get lost on the way. The challenge of capturing and maintaining huge volumes of knowledge requires new ways of knowledge acquisition as found in [151]; namely, on approaches that rely on the contributions of many rather than the expertise of a few. This factor of success was also reported by studies in the medical domain [156].

A closer investigation identified the lack of shared common knowledge in the manufacturing domain as a core problem. This problem is bundled in RMF2 and the insights gained here are consistent with insights gained by [153], who found that an interoperable model is needed to represent the diverse collection of knowledge for environmental monitoring. A related Resource Management Factor RMF3 is the need for a standard vocabulary specified as basis for implementing a knowledge-based RMS in the manufacturing domain. Software tools and experts of the manufacturing domain require a shared terminology and syntax in order to efficiently and effectively inter-operate. Studies of [72] found that for

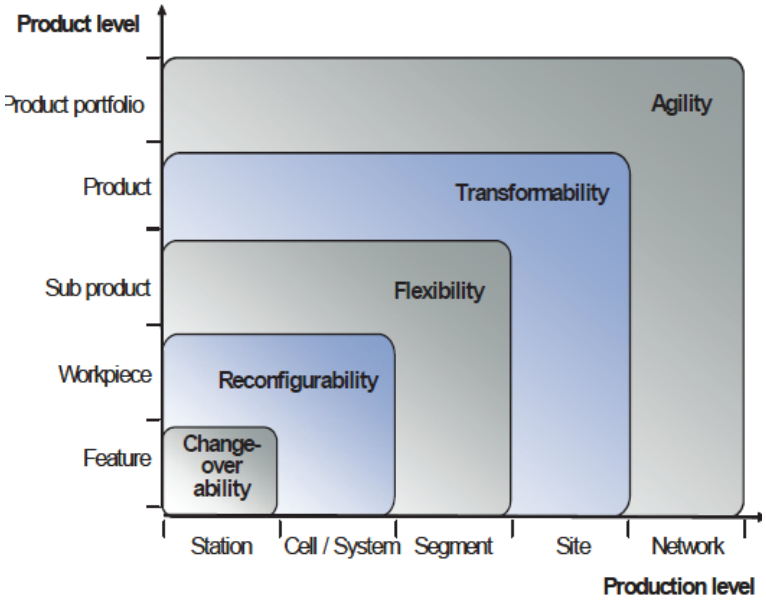


Figure 5.1.: Classes of Plant Reconfigurability [158]

an effective human-to-human, human-to-machine, and machine-to-machine communication in the manufacturing area, a common understanding of business and manufacturing-related terms is indispensable. According to [81], a well-defined generic manufacturing taxonomy and axioms are required that can be accepted by all participating engineers to make design knowledge effectively accessible. This statement corresponds to factor RMF4 suggesting that a generic plant description would allow a broader acceptance of modern RMSs. An interesting study proposed by [111] discusses as well deficits of knowledge-based approaches for the engineering of automation systems and identifies challenges for knowledge-based approaches in the industry automation domain. One challenge identified here was that a generic and simple formalism is needed to describe and exchange knowledge for the engineering of automation systems. This finding confirms the importance of RMFs RMF2 and RMF4.

Reconfigurability is a key enabler for meeting the challenges of a global market as found by the aforementioned studies in RMF5; this can be confirmed by [158] and [78]. More specifically, reconfigurability on all levels of an industrial plant need to be tackled as presented in Figure 5.1.

RMF6 showed the importance of high sampling rates for detecting electricity peaks of machines. This factor could be confirmed by studies on energy management systems in [67]. Their study suggests that sampling rates approaching tens of kHz or even MHz may be needed to faithfully capture electricity peaks in industrial plants.

A central Resource Management Factor for RMS concerning the introduction in the mar-

ket is the reusability of such systems. As stated in RMF7, high reusability allows to save costs at implementation time. This has also been shown in studies conducted by [18] where code reuse allowed to cut automation project costs by more than 30%. Moreover, available standards should be evaluated according to their cost-benefit ratio. [23] pointed out the importance to evaluate energy management standards regarding their cost-benefit ratio and impact of implementation.

A detailed study conducted by [130] identified Energy Management Factors concerning the measurement and monitoring of energy efficiency in the pulp and paper industry. More precisely, they state that energy efficiency is influenced by numerous internal and external variables and their dynamics such as ambient conditions, the occurrence of breaks and shutdowns, the production rate, the quality of the materials, etc. These variables need to be analyzed by complex algorithms. Energy efficiency monitoring is an important part of resource monitoring and thus RMF8 can be confirmed based on this study.

In conclusion, all of the identified RMFs could be confirmed by at least one study. The identified factors need to be tackled by defining a systematic method and requirements for specifying a knowledge-based RMS. An overview of all RMFs and correlating studies is shown in Table 5.1.

Resource Management Factor	**Similar studies**
RMF1 (knowledge-acquisition bottleneck)	Wagner [151], Wennerberg et al. [156]
RMF2 (shared common models)	Wang et al. [153], Runde et al. [111]
RMF3 (standard vocabulary)	L et al. [72]
RMF4 (generic plant description)	Lin and Harding [81], Runde et al. [111]
RMF5 (high degree of reconfigurability)	Wiendahl et al. [158], Legat et al. [78]
RMF6 (high sampling rates)	Jiang [67]
RMF7 (implementation costs)	Brandi [18], Bunse et al. [23]
RMF8 (complexity of algorithms)	Sivill et al. [130]

Table 5.1.: Overview of all RMFs and related studies

6. Knowledge Engineering Methodology

Based on the Resource Management Factors identified in the previously mentioned studies, a systematic sequence of tasks for definition of a knowledge-based RMS was defined. This sequence of tasks is represented by a knowledge engineering methodology that allows to gather and represent the knowledge needed for resource monitoring [3]. Plant engineering experts have to stick to this knowledge engineering methodology presented in Figure 6.1, when implementing a knowledge-based RMS in alignment to the Resource Management Factors. Four main tasks were specified in the subsequent sections for implementing the RMS:

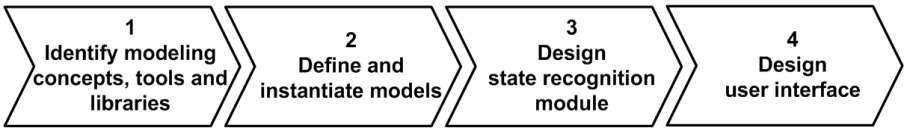


Figure 6.1.: Knowledge engineering methodology

Identify modeling concepts, tools and libraries: Several tools and documents are required to support plant engineers in the plant design process. All important terms, objects and relationships required for this process have to be covered by the identified modeling concepts. Then, the modeling formats and libraries that correspond best to the identified modeling concepts have to be selected.

Define and instantiate models: For effective knowledge reuse, all relevant modeling concepts are stored in a model library. This library contains generic and specific models which are extensible and can be instantiated for specific facilities. Plant engineers can navigate and search the library and the models can be used for automated reasoning.

Design state recognition module: The core task of a monitoring system is to compute the monitoring states of the plant devices and present them to the operator. Systems that actively control the monitored system are not considered, only systems that provide decision support for the operator. In our methodology, a state recognition module defined by the plant engineers, operators and device manufacturers is necessary to compute monitoring states.

Design user interface: The operator interacts with the run time monitoring system via user interfaces. Three kinds of user interaction should be supported by a monitoring system: (1) periodically computed monitoring results, (2) computing and reporting of alarms, (3) user queries.

Analyzing the methodology in more detail, a special requirement for the industrial do-

main was discovered, which influences the entire methodology. As mentioned in Section 4, a challenge of the industrial domain is that in addition to the generic domain models containing abstract engineering and automation concepts, a level of facility-specific description is required to add specific knowledge about a certain facility. Both levels require several steps with input from multiple knowledge sources. Furthermore, the resulting output of the steps can be reused by future facility-specific monitoring systems. The following sections describe the individual tasks on the general and facility-specific levels in more detail.

6.1. Identify Modeling Concepts, Tools and Libraries

First of all, the most appropriate modeling concepts for the industrial domain have to be identified based on the requirements, a literature review and the input of domain experts.

For example, AutomationML [41] provides concepts to model a facility, e.g. *component* (called *resource*), *process*, *interface* and *role*. The AVILUS project provides concepts for the life cycle of modern production plant engineering and service tasks [20]. Web Services provide concepts like *service* and *operation*. In this methodology, the domain models contain all identified concepts. Based on the requirements, the plant engineer chooses specific concepts for the facility-specific model. Candidates which could be selected from existing model formats are industrial ontologies, such as SWEET [108], or industrial modeling formats such as CAEX [116].

Given the initially identified challenges, our main criteria for this step were:

- The model format has to be vendor-independent.
- The models are generic, not restricted to a specific area.
- Experts of different areas can extend the models in the scope of their expertise.
- A model can refer to concepts contained in another model.
- The meaning of terms can be described formally. This makes it possible to map concepts from one model onto concepts of another model, e.g. *contains* and *has part* are *aggregations*.
- Facility-specific models can be built based on the generic domain models by instantiating the domain classes.

Based on these criteria, existing models and modeling formats were evaluated. First of all, industrial ontologies were considered. In other domains, such as the medical domain, this step is simple, since high-quality medical ontologies have been developed over the years as a result of joint efforts of knowledge engineers and health care experts [156]. Whereas in the industrial domain, only few existing ontologies were proposed by academia. Nevertheless, several ontologies such as SensorML [84], SWEET and the Process Specification Language Ontology PSL [37] could be identified. PSL was chosen to model plant

processes, because all other identified ontologies were restricted to specific areas and not generic enough for the purposes of this work.

Second, industrial modeling formats were compared. There are commercial industrial software tools such as Siemens COMOS [123] or PLM (product lifecyle management)[126] solutions, which capture and manage all plant-related information. But these formats are vendor-dependent, and thus not suitable for general domain models.

There is currently one area- and vendor-independent exchange format for plant engineering: AutomationML and its related formats CAEX, COLLADA and PLCopen XML [41]. Several industrial applications, e.g. an automatic configuration of a production monitoring and control system, have been built based on CAEX as exchange format [58]. But the usage of a data exchange format such as AutomationML requires tool support. There are some dedicated tools such as the AutomationML Editor of Zühlke Engineering AG [41] or the CAEX tool suite [115], but currently they only offer basic features like navigation ability. Furthermore, there is no mechanism to coordinate classes across several companies.

Another upcoming standard of the industrial domain is OPC UA, a set of specifications for process control and automation system inter-connectivity. It can also be used to define information models in the automation domain as pointed out in [159]. A special feature of OPC UA is that it integrates an Object Model, which provides type definitions for objects and their devices. On top of this object model, a Base Information Model is available, which allows to define structure, behavior and semantics of the objects. This approach meets most of our criteria, but OPC UA does not define a data format to store the models, only to query the devices at run time, and the models are not extensible.

The next step is to collect libraries with detailed device information either provided by product catalogs or industrial standards like IEC 61360-2 [1] or ISO 13584 [2]. An example for such a library entry is “the motor XYZ is produced by Siemens, has the initial rotational speed of 3 rpm, has an energy efficiency range of 60%-82%”.

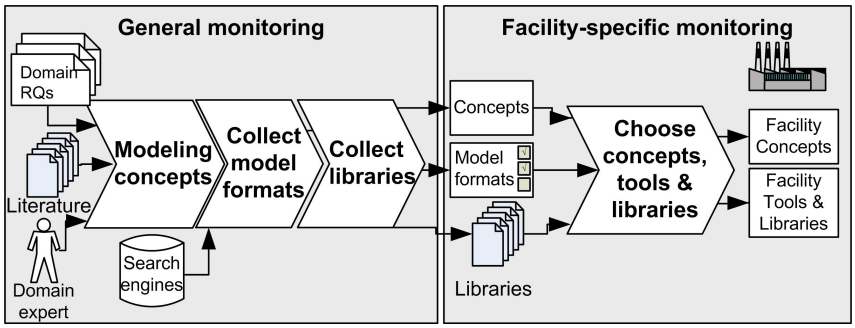


Figure 6.2.: Identify modeling concepts, tools and libraries

[1] IEC 61360-2 (2012):”Standard data element types with associated classification scheme for electric components”

[2] ISO 13584 (2006):”Industrial automation systems and integration - Parts library”

On the facility level, the results of the general monitoring level are reused as shown in Figure 6.2. Depending on facility-specific constraints, e.g. the modeling tools and device libraries used by the plant owner, the plant engineer chooses appropriate concepts, tools and libraries for the facility.

6.2. Define and Instantiate Models

To define facility-independent partial plant models, several parties are involved which cover different areas of expertise. The resulting partial plant models may contain partially redundant information, synonymous terms and homonym terms. These models have to be connected and mapped to each other to build consistent models with well-defined semantics. All concepts, tools and libraries identified in task 1 form the basis for specifying these models.

The partial plant models are stored in a model library, so that they can be reused. As described in the next section, the aim is to define models that can be processed by a state recognition module, reducing the development time and the overall system costs. Extensibility and compatibility are important characteristic of the library. To guarantee these characteristics, experts with different background knowledge should have the possibility to either add their additional concepts to the library or match their concepts to existing library concepts in an iterative way.

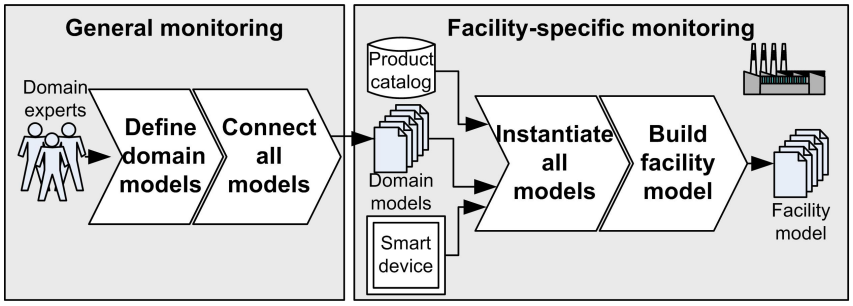


Figure 6.3.: Define and instantiate models

During the plant design process, the plant engineer can download device models from the manufacturers' product catalogs as shown in Figure 6.3. On smart devices, the manufacturer can attach all device-related information directly on the device using Digital Object Memories (DOMs) as proposed in [120, 121], e.g. serial number, production date or maintenance log. When the devices are mounted, the detailed device models are integrated into the facility-specific models, and the relations between device instances can be added, e.g. "the motor m_1 has the relationship *has part* to its sub-component, the temperature sensor ts_1".

The information contained in the facility models is available for services in the sense of

service-oriented architecture (SOA). Facility- and company-independent services such as monitoring of resource consumption, reporting or diagnosis are listed in a service catalog together with semantic descriptions. With this approach, results of one service can be offered to other services via semantic interfaces, e.g. monitoring results can be used by diagnostics or optimization systems.

6.3. Design State Recognition Module

The state recognition module (SRM) is the central part of the monitoring system. It receives the signals from the plant sensors, uses the models to annotate the sensor data semantically, computes the monitoring states of components and groups of components, and provides the states and the annotated data to other services.

In our methodology, rules executed by rule engines are used and classified in two kinds of rules: rules that infer states of single components and composite rules that infer states of component groups consisting of several components.

In the first step, the knowledge engineer collects representative rules from the domain experts. With these exemplary rules, he makes a pre-selection of appropriate rule engines. For these rule engines, he defines general monitoring rules for different use cases, e.g. the monitoring of resources.

For example, the model may specify that sensor s_1 measures the temperature of motor m_1. If s_1 reports a value of 85, the SRM annotates this as "motor m_1 has a temperature of 85 °C". The SRM executes the rule "if the temperature of motor m_1 is higher than 80 °C, then m_1 has the state *error*" to infer the state of m_1, and the composite rule "if one of the motors in the motor group G has the state *error* then the state of G is *error*" to infer the state of G. Then the SRM provides the inferred states and the annotated data to other services.

The SRM must support "pull" communication at the field level of the automation pyramid, where it queries the sensors periodically and reports alarms in the case of erroneous system behavior or failure. The SRM must also support "push" communication where smart components only report the events and alarms that the SRM has subscribed to leading to a reduction of the data transfer volume.

The facility-specific rules are specified by plant engineer and knowledge engineer. First they select a rule engine, or several rule engines for different tasks (e.g. rules vs. composite rules). Then they execute consistency checks on the models to discover invalid relationships between objects, e.g. "the relationship *monitors* can only be defined between two components, not between two processes".

Then they specify rules for the components and component groups to infer the state. For example a facility may require four levels of severity for the rotation speed, torque and temperature of its motors: *normal*, *warning*, *error*, *critical error*. Then the motor m_1 will have four state categories for the temperature, e.g. *warning: temperature of* m_1 *too high*, and similarly for rotation speed and torque.

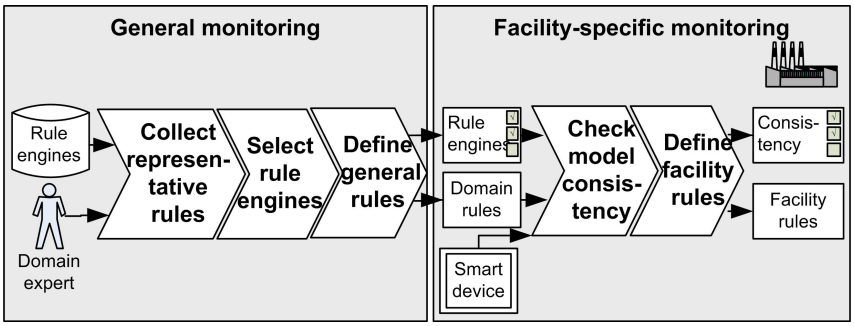


Figure 6.4.: Design state recognition module

Some of the states and rules may be defined by the component manufacturer. Smart components can execute the rules on the component, otherwise they will be executed by the SRM. Other states and rules will be facility-specific and must be formulated according to the general monitoring rules as shown in Figure 6.4. Furthermore, the engineers define composite rules which infer the states of composite components and the entire facility.

6.4. Design User Interface

Three kinds of user interaction with the RMS are supported: (1) periodically computing and displaying monitoring results, e.g. “the average pressurized air used in the last hour”, (2) reporting alarms and events that are computed automatically by the system, e.g. “Error at 15:01:30: Oil consumption of motor m_1 is too high”, (3) the operator can query the system, e.g. “What is the current energy consumption of the entire plant?”.

7. Requirements for Resource Monitoring Systems

In Chapter 4, eight Resource Management Factors crucial for the successful implementation of a knowledge-based RMS were identified by means of empirical studies. To guide the definition of a system in alignment to the identified Resource Management Factors, a transfer of these factors on a set of requirements for technical realization is helpful. This chapter provides a requirements analysis for creating and categorizing a comprehensive list of requirements necessary for defining an RMS in a modern manufacturing environment. A distinction between two kinds of requirements is made: 1) *core requirements*, by definition, are pertinent to the overall functionality of the monitoring system and correspond to the functional system requirements, 2) *outer core requirements* are pertinent to some specific technical or usability demands.

7.1. Core Requirements

Requirement 1 (application independence):

Current RMSs are typically either plant-specific software solutions or they focus on monitoring of specific equipments (e.g. bearings, pumps or motors) [137] and therefore require significant manual investment and effort for implementation. To reduce this effort, an application-independent approach is required that can be reused for various kinds of plants. Furthermore, plant-independent solutions focus usually on specific branches of industry (e.g. metal industry, paper industry) within specific manufacturing engineering areas (e.g. process manufacturing, discrete manufacturing).

In order to guarantee a maximum reusage of knowledge across different industrial branches, an RMS has to be generic and universally applicable independent of the branch of industry and the manufacturing engineering area.

Requirement 2 (reconfigurability):

Modern industrial plants are fast changing environments that require steady adaptations. Several evolutions in manufacturing, such as the introduction of new manufacturing materials and technologies, the development of new products, as well as the increasing

need for responsiveness, adaptability and agility require a higher reconfigurability of RMSs [85].

> Users of the RMS should be able to adapt models of the plant that are needed for monitoring, extend the models with additional concepts, reuse the device libraries and rules of existing plants for new plants and define their own plant-specific rules with low effort.

Requirement 3 (integration):

According to [32], diagnosis systems can be improved by using knowledge about the system's structure and processes, such as (i) product flow through a plant, (ii) energy flow between plant components, etc. This is also valid for monitoring systems as shown in [6]. For the monitoring of resources in industrial plants, context information is relevant too. The computation of states, for example, depends on the production context: produce "with lowest energy consumption" or "in minimum delivery time". Thus, RMSs do not only require a measurement of sensor data, but also various kinds of discipline-specific expert knowledge about the plant that has to be stored in partial plant models, such as process, structural or context models, to give the operators background information about monitoring results computed by rules according to the paradigm called model-integrated mechatronics (MIM) specified in [139]. An architecture is needed that promotes model integration and significantly decreases development and validation time.

> The users have to be able to integrate their specific models in a collaborative way in a knowledge-base that can be processed by the monitoring rules.

7.2. Outer-Core Requirements - Technical

Requirement 4 (algorithmic and logical operations):

Engineers usually describe characteristics of devices and facilities by mathematical expressions and logical operators. These characteristics need to be stored in the knowledge-base. The sensor data and the knowledge in the knowledge-base are then processed by the rules.

> The RMS should support methods to perform complex algorithmic and logical operations with rules to derive new facts from the real-time input data or identify trends in data sets.

Requirement 5 (time series analysis):

The RMS has to observe dynamical processes, e.g. to monitor the correct sequence of processes. An appropriate means for this analysis are time series analysis tools, e.g. Complex Event Processing rule engines.

> Time series analysis must be supported for monitoring of dynamical plant processes, especially for analyzing time correlation of plant engineering data.

7.3. Outer-Core Requirements - Usability

Requirement 6 (modeling patterns & libraries):

Knowledge-based technologies are usually well-suited to face the challenges described in the previous requirements, but the engineering of such knowledge-based models and rules is usually a task of knowledge engineering experts because it requires specialized know-how [47]. This limitation impedes plant engineering experts to construct or adapt such systems since they are often not familiar with knowledge-based technologies.

> Engineering experts must be supported in their modeling task by predefined modeling patterns for the metalevels and libraries of devices or processes.

Requirement 7 (consistency checks):

During the engineering process of RMSs various experts with different backgrounds are involved, such as electrical engineers, mechanical engineers, etc. In the course of engineering plant models and monitoring rules, various inconsistencies might be introduced [45], e.g. distinct elements may accidentally have the same identifier. To compute states based on a consistent plant model, these inconsistencies have to be identified by automatic consistency checks of models that validate them against predefined modeling patters. To avoid the propagation of inconsistencies and modeling mistakes, each and every task has to be tested for validity and consistency.

> Engineering experts must be able to specify their own plant-specific constraints in addition to predefined constraints which are automatically tested for validity and consistency.

Requirement 8 (navigation):

According to [54], an issue with knowledge-based approaches is the development effort and the extensive process knowledge needed for the reasoning task. Since only engineering experts have extensive knowledge about how to monitor the plant and detect

faulty behavior, we need to ensure that they are guided during the engineering phase of the RMS by means of advanced navigation and search features.

The RMS has to support users in navigating within different aspects of plant knowledge and the system must allow the users to search in plant models and monitoring rules for specific knowledge.

Requirement 9 (maintenance of rules):

Typically, several domain experts supported by the knowledge engineering experts are involved in the engineering phase of RMSs and generate a huge amount of monitoring rules. Difficulties arise, however, when contradicting or redundant rules are generated or when the number of rules exceeds a certain limit that the experts cannot cope with. Hence, the rules need to be structured in a hierarchical way to ease their maintenance.

The users have to be supported in structuring rules and identifying inconsistencies in the rule base.

7.4. Relation between Resource Management Factors and Requirements

The requirements defined in this chapter were derived from the RMFs defined in Chapter 4. The relations between them are shown in Figure 7.1. The arrows in the Figure represent a "derived from" relationship between RMFs and requirements. All requirements were derived from several RMFs.

For example, RMF 2 showed that experts have to share common models for an industrial plant to allow effective resource monitoring. This factor implies a strong independency of the application (RQ1) to share models of different plants, the possibility to integrate different models in one monitoring system (RQ3) and the need to navigate between the models (RQ8).

The experts involved in the empirical studies were asked to verify the validity and completeness of the requirements based on Figure 7.1. They confirmed that the requirements for the RMS are valid and that the broad spectrum of RMFs is covered by the identified system requirements. Furthermore, most of the system requirements can be verified by the literature review presented in Chapter 3.

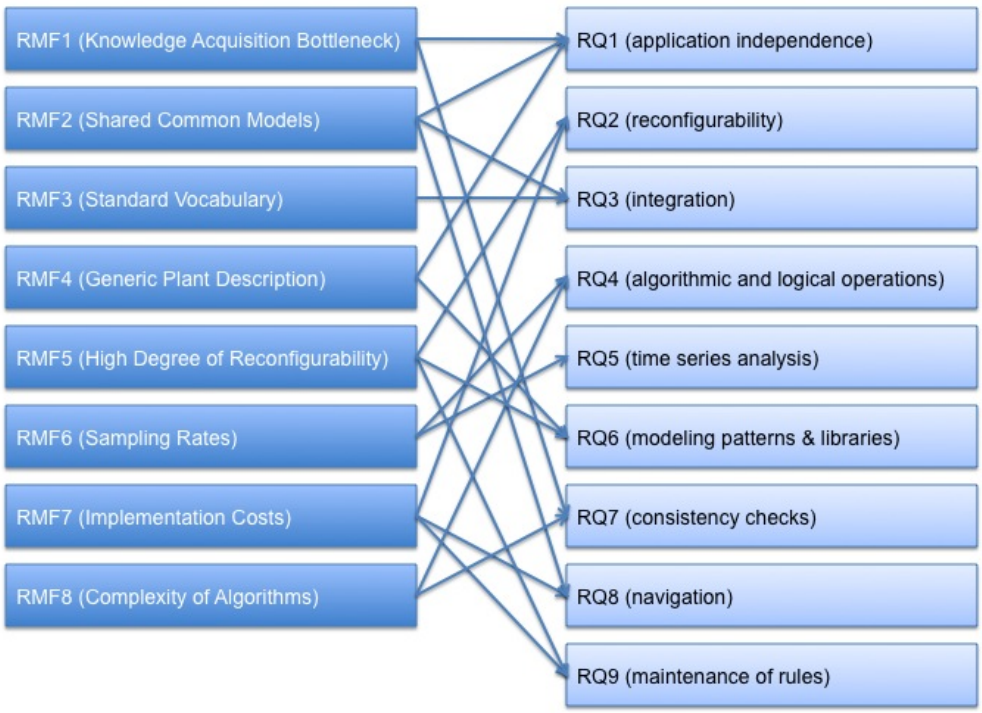


Figure 7.1.: Relation between RMF's and Requirements

Part III.

Knowledge-Based Resource Monitoring Approach

8. Plant Engineering Models

The construction of a new industrial plant is a complex process where various design decisions have to be made within multiple plant engineering steps. Within this process, the plant engineers have to decide which hardware components have to be arranged to an assembly, how these components are connected via mechanical or electrical connections, which processes are executed by the components, what kind of products are produced, which resources are consumed by the components (e.g. energy, water) and in what context the plant and their components reside (e.g. 'produce with low energy consumption' or 'produce in minimum delivery time'). These decisions involve several engineering disciplines, in addition to different engineering tools and standards of the heterogeneous industrial tool landscape. As pointed out in Chapter 4, a generic plant description (RMF4) based on a standard vocabulary (RMF3) and common plant models (RMF2) are needed for integrating the knowledge involved in these decisions in the RMS.

A further issue is that experts with different roles have role-specific demands on a generic plant description. Current research usually focuses on defining partial plant models that correspond to the manufacturing goals from the viewpoint of one expert of a specific domain [160]. However, the joint consideration of the various perspectives of all domain experts enables knowledge engineers to develop manufacturing systems, and specifically RMSs, with broader functionality.

The aim of this chapter is thus to define the theoretical basis for a generic plant description to tackle these issues. This generic plant description builds on a ***conceptual modeling approach*** that makes use of a ***concept model*** to provide a common way for engineering experts to express knowledge about their plants. The main purpose of this approach is to specify the usage of the provided concept model to allow for the integration of several partial models defined by various domain experts. The conceptual modeling approach represented in the following sections is based on a four-level metamodel architecture used to specify generic partial plant models which can be instantiated for specific facilities. Once instantiated, the models can be stored in a knowledge-base and used by the RMS. Such a generic plant descriptions allows the plant engineers, who construct the RMS to navigate within the knowledge-base and search for specific information.

In Section 8.1, the conceptual modeling approach and its usage is described. Examples for partial plant models such as the structure model and the process model specified by the author of this thesis are given in Section 8.2 and 8.3 respectively. An evaluation of the described approach is reported in Chapter 9.

8.1. Conceptual Modeling Approach

As stated at the beginning of this chapter, an approach is needed that allows plant engineering experts for integrating different plant aspects in a collaborative way in a generic plant model. The goal of this section is to define a conceptual monitoring approach to specify such a generic plant model according to the specification defined in [2]. First, fundamental terms of the conceptual modeling approach are listed in Section 8.1.1. Then, the conceptual modeling approach is described in detail in Section 8.1.2 and a concept model is introduced that provides a common way for domain experts to express knowledge about an industrial plant in a machine-readable format in Section 8.1.3. Instructions for using the concept model in practice are given in Section 8.1.4.

8.1.1. Fundamental Terms of the Conceptual Modeling Approach

This section lists fundamental terms required to establish an understanding of the conceptual modeling approach described in the following sections. The terms are listed in alphabetic order.

Class

In the concept model, the term class is used when attributes and relationships of a type are valid for all its instances. For example, consider Tolkien's book Lord of the Rings with the ISBN *9780261102415* defined as follows:

```
class Lord_of_the_Rings_9780261102415 {
  attribute ISBN = "9780261102415"
  attribute title = "The Lord of The Rings"
  attribute author = "J.R.R. Tolkien" }
```

Every instance of this class will have the same attributes for ISBN, title and author, so it's appropriate to use a class.

- **Comparison to other languages and standards:**
  - **OWL:** OWL uses class in a related but different sense, since attribute values are not specified specifically for classes.
  - **UML:** The usage of the term class here is similar to the definition of class in UML.

Datatype

A ***Datatype*** is a simple Type like `integer` or `string`. Its elements are called *values*. In contrast to instances, values do not have an identity.

- **Comparison to other languages and standards**
  - **OWL:** OWL also refers to datatypes, see `owl:DatatypeProperty`.
  - **Programming languages:** Built-in types in C++, C# and Java are datatypes.

Instance

An ***instance*** is an element in a model specified by its type. In the concept model, ***datavalues*** are not considered to be instances. The difference between datavalues and instances is that instances have an identity that is independent of their attributes.

For example, the motor "m1" is an instance of the type "Siemens Motor 1FK7032-5AK71".

An instance can have zero, one or more types. In this sense, ***multiple classification*** is allowed. For example, a person may be an Employee, a Manager and a Patient at the same time.

- **Comparison to other languages and standards:**
  - In other languages and standards, instance may include values.
  - An instance often corresponds to objects in OWL, UML, C++, C# and Java. But there are objects like "Date" that are considered to be values, not instances.

Multiple Classification

Multiple classification means that an instance can have several types as specified in [86]. It's the opposite of ***single classification*** and different from ***multiple inheritance***.

Multiple Inheritance

Multiple inheritance means that a type can have several supertypes [62]. It is the opposite of ***single inheritance*** and different from ***multiple classification***.

Template

A ***template*** is a kind of type. It is similar to a class, but instantiated by **copying**. Instances of a template can change the value of their class attributes. An example is a *Hardware Component Template* in the structure model as shown in Figure 8.1. Consider a motor *m1* with the *Template Motor1FK7*. The attribute of the motors can change, e.g. the *maximum energy efficiency* decreases over time. Thus, we need templates here and not classes, where the *maximum energy efficiency* has to be identical for every class and its instances.

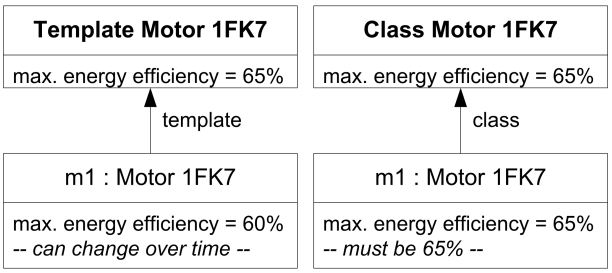


Figure 8.1.: Comparison of class and template attributes - class attributes are constant, whereas template attributes can change

Type

All main instances of the concept model are typed. A distinction is made between three different kinds of types: classes, templates and datatypes as depicted in Figure 8.2.

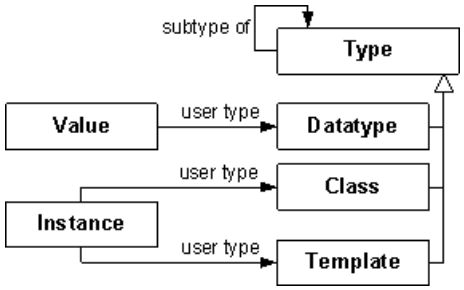


Figure 8.2.: UML class diagram of different kinds of types

A type is a description of a set of similar things. Consider the following examples:

- the numbers 1 and 2 have the type integer
- the text "hello" has the type string
- the motor "m1" has the type "Siemens Motor 1FK7032-5AK71"

Type is a modeling concept that is independent of languages and standards like CAEX, OWL, UML, etc. A type may have several ***supertypes*** (connected with *subtype of)*, since multiple inheritance is allowed.

Quantity

In the simplest case, a ***quantity*** is a number with a unit, for example “10 kg”. Quantities may be vectors, e.g. an acceleration along 3 axes: $a = (1\frac{m^2}{s}, 2\frac{m^2}{s}, 3\frac{m^2}{s})$

8.1.2. Detailed Description of the Conceptual Modeling Approach

The conceptual modeling approach described in the following chapter makes use of a generic plant model, named concept model, to provide a common way for engineering experts to express knowledge about an industrial plant. The main task of this approach is to specify the usage of the concept model to allow for the integration of several partial plant models defined by various domain experts.

Modeling Levels of the Concept Model

Three different roles of domain experts in the manufacturing areas could be identified as target users of the ***concept model*** as shown in Figure 8.3:

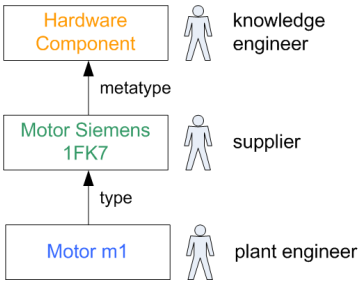


Figure 8.3.: Three expert roles

1) the *knowledge engineers* specify the partial plant models for various discipline-specific aspects of a plant in collaboration with the engineering experts (e.g. a concept “Hardware Component” stored in the structure model),

2) the *suppliers* (manufacturer, third party supplier or service supplier) specify abstract information about their equipment (e.g. the manufacturer Siemens specifies characteristics about the “Motor Siemens 1FK7”),

3) the *plant engineers* (plant owner, engineering expert or operator) construct, operate and maintain the plant models (e.g. about a specific “Motor m1” in their plant).

Summarizing, three levels of users were identified on three different levels of abstraction. In software engineering, entities on the lowest level of abstraction are called “”objects” (or instances), on a higher level of abstraction entities are named “types” and the entities on the highest level of abstraction are also called “metatypes”.

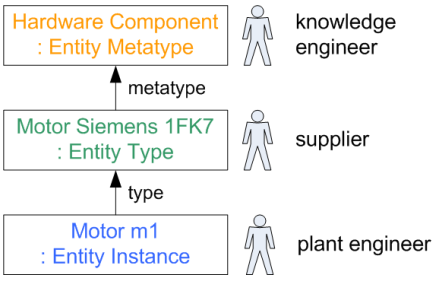


Figure 8.4.: Three levels of abstraction

When annotating the entities presented in Figure 8.3 with such a semantic software engineering description, this results in the UML diagram presented in Figure 8.4. Remarkable is that the entity *HW Component* is associated with the class ***Entity Metatype***, the entity *Motor Siemens 1FK7* is associated with the class ***Entity Type*** and the entity *m1* is associated with the class ***Entity Instance***. This is a bit different from "the usual usage of UML", where *m1* is an object (or instance), *Motor Siemens 1FK7* is a class, and *HW Component* is a stereotype as depicted in Figure 8.5.

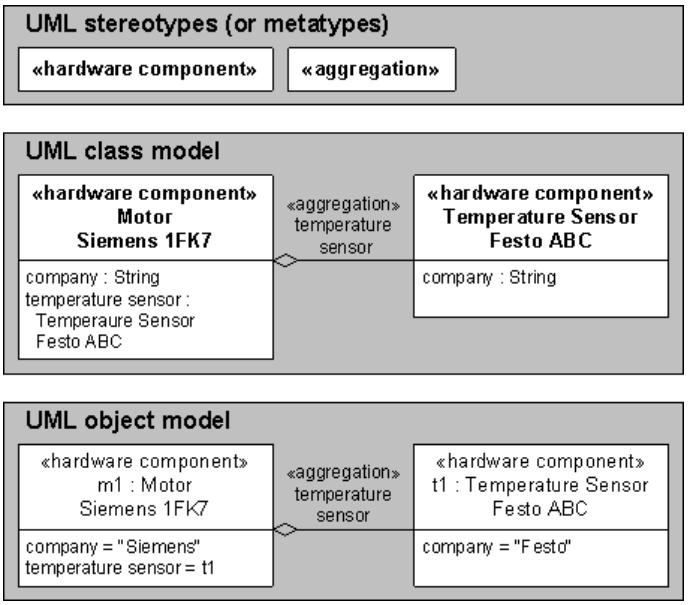


Figure 8.5.: The three modeling levels in UML

The idea of the conceptual modeling approach is to support the identified roles by a four layered metamodel architecture to structure the plant knowledge. This metamodel architecture is also called ***model-driven architecture*** and related to multiple standards or model formats defined by the Object Management Group (OMG) [98], including the Unified Modeling Language (UML), XML Metadata Interchange (XMI)[1], the Software Process Engineering Metamodel[2], and the Common Warehouse Metamodel (CWM)[3]. Note that the term "architecture" in MDA does not refer to the architecture of the system being modeled, but rather to the architecture of the standards and model forms that serve as the technology basis for MDA. The most prominent standard of OMG is the Meta Object Facility (MOF) Core Specification 1.4 [98]. The key modeling concepts of the MOF are *type* and *instance*, and the ability to navigate from an instance to its meta object (its type). Such a MDA for the representation of layers allows engineers to participate in the modeling process in a collaborative manner and to ensure a formal representation of the expert knowledge as shown in [75]. The conceptual modeling approach is grounded in MOF, in the sense that it is defined in terms of the MOF meta-metamodel. Based on these findings, the following levels can be distinguished:

- **Level M3 (Concept):** The concept model specifies general concepts that can be used by the software engineers to define their partial plant models. These concepts

[1] ISO/IEC 19503 (2005): "Information technology – XML Metadata Interchange (XMI)"
[2] OMG standard (2008): "Software & Systems Process Engineering Metamodel specification Version 2.0"
[3] OMG standard (2003): "Common Warehouse Metamodel, v1.1"

Concept	Informal Definition
Entity	An element in an industrial plant that can be classified and has relations to other entities. For example, the structure model defines the entities *Hardware Component*, *Role Class* and *Interface*.
Relationship	A specification that connects two or more Entities, the source and the target. Visually, it is an arrow from the source to the target entity. A relationship used to define a containment hierarchy is “has part”.
Attribute	A specification that defines characteristics of an Entity or a Relationship, e.g. a *Hardware Component Template* can have attributes like ’input power = 750 kW’ and ’motor type = asynchronous motor’. Attributes are primarily intended for numbers, quantities or strings. Both, Entities and Relationships can have Attributes.

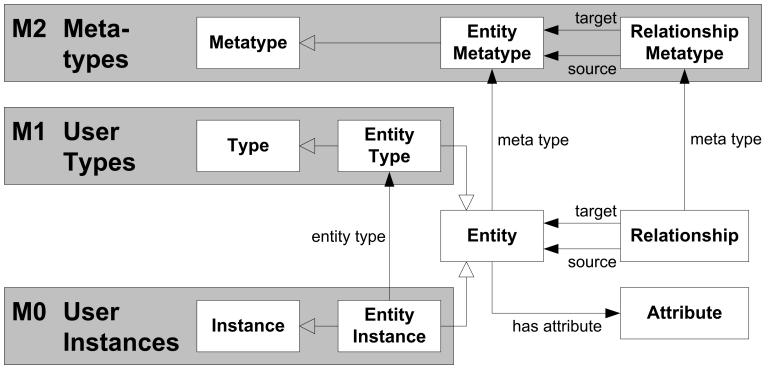


Figure 8.6.: Concepts and their connections introduced by the concept model

were chosen in accordance to principles of object-oriented design since software and especially knowledge engineers are familiar with them. The three basic concepts used for modeling at this level are presented in Figure 8.6 and Table 8.1.2. A detailed description of the concepts is given in chapter 8.1.3.

- **Level M2 (Metatype):** The *knowledge engineers* who specify the partial plant models on this level describe or discuss specific concepts and terms with experts of different engineering disciplines. The resulting model is called ***metamodel*** and can be used by level M1 and M0. The elements that are used by the authors and called ***metatypes***, e.g. an engineer defines the metatype *Hardware Component*. An example for such a partial plant model is the structure model as defined in more detail in Section 8.2.

- **Level M1 (User Type):** The suppliers, e.g. the manufacturers of components, store general information on types in their model such as specifications of their products in a product catalog. The resulting model of the M1 level is called ***user type model*** and used by the M0 level. For example “Siemens” describes the *Motor Siemens 1FK7* and its specification details in a data sheet.
- **Level M0 (User Instance):** On this level, plant engineers define a concrete industrial plant with instances of the types of level M1. The resulting model is called ***user instance model***. The plant engineer specifies this model when he installs components of the suppliers in his plant and stores all component-specific information and the relations between the components, e.g. *Motor m1* is an instance of *Motor Siemens 1FK7*.

Practically, the concept model corresponds to the M3 layer of the MOF, the metamodel to the M2 layer, the type model to the M1 layer and the instance model to the M0 layer as shown in Figure 8.7.

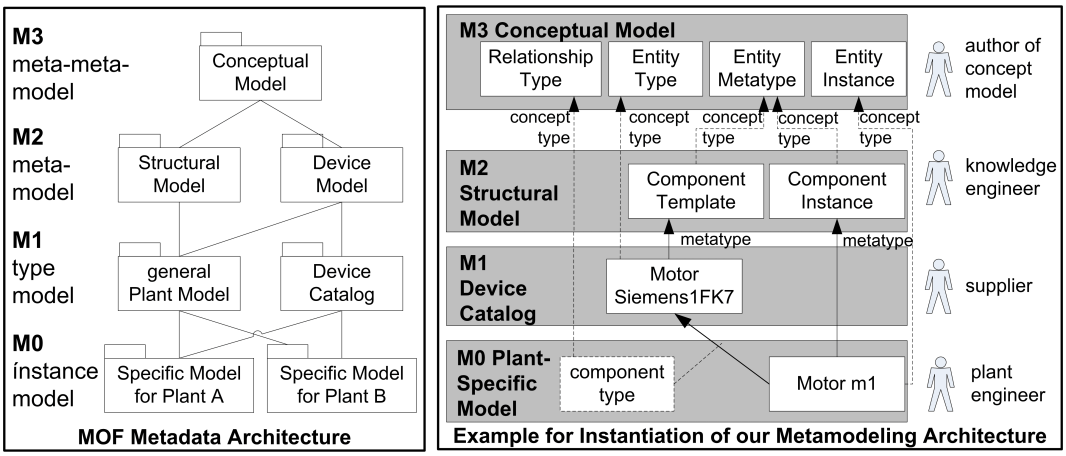


Figure 8.7.: Modeling levels of the concept model in correspondence to the MOF modeling levels as defined in [98]

To connect the different levels, relationships between the levels are required. Therefore, relationships from M0 to M1 level are defined which are named *entity, attribute or relationship type* and additionally a *meta type* relationship that refers from level M0 or M1 to the M2 level. The full concept model with all its concepts and relationships between them is presented in 8.8. Furthermore, all elements need a reference to the concept model, which is named *concept type* and required to distinguish the different levels of all elements. Additionally, these levels are needed during implementation, since this distinction allows for storing the elements from the M0, M1 and M2 levels in an element of the M3 classes.

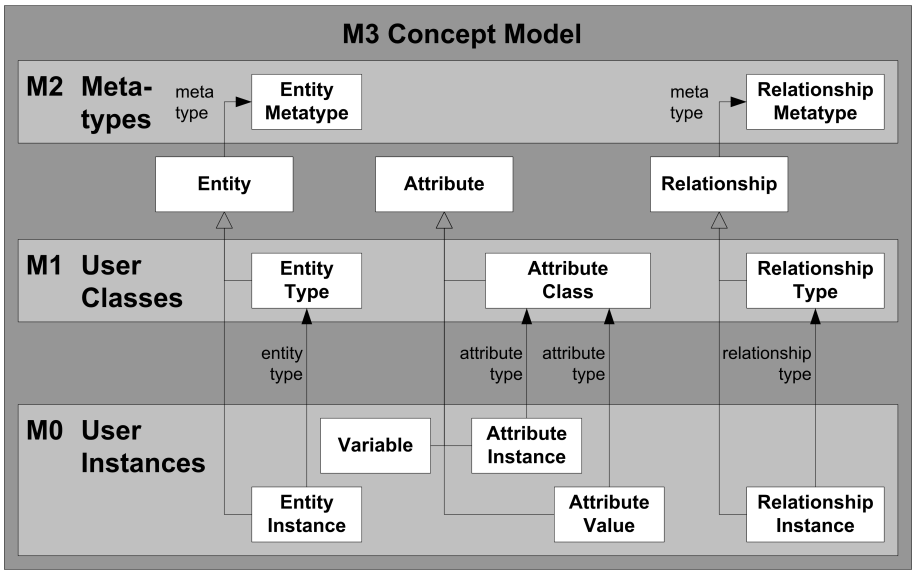


Figure 8.8.: Concept model with all levels and concepts

Using this four layered architecture, the concepts Entity and Relationship defined on the M3 level, are used on all four levels by the domain experts to build their models. The definition of Attributes is limited to the M1 and M0 level since they are specific for every component and can thus only be defined by the suppliers or plant engineers not by the authors of the metamodels.

An advanced example of the concept model which includes all levels and relationships between the levels is displayed in Figure 8.9, it includes all type relationships:

- *user type* (colored orange, from M0 to M1), also used on level M2 to define domain and range of links on level M0/M1
- *meta type* (colored purple, from M0 or M1 to M2)
- *concept type* (colored green, from M0, M1 or M2 to M3)

Specific details of the concept model need to be considered prior to its usage:

- To instantiate an entity or relationship from M2, call the class that "level" points to. This defines in which level the element or link will be.
- The red relationship "level" is needed to know in which level (M0, M1, M0 to M0, M0 to M1, M1 to M1) to instantiate a concept.
- The classes on M3 level can be used for two things:
  1. to distinguish levels M0, M1 and M2 by means of the relationship *concept type*

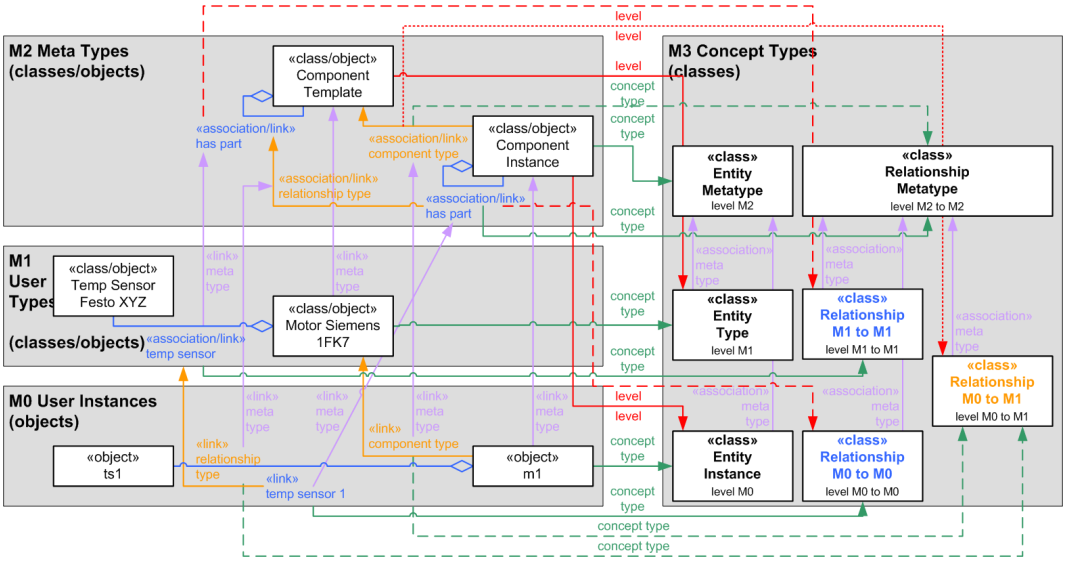


Figure 8.9.: Four-layered model architecture with all type relationships

2. when implementing the entire concept model, an element of the M3 classes stores the elements from the M0, M1 and M2 levels

The concept model also specifies basic modeling characteristics to impose constraints on inheritance or classification of modeling entities. It allows for multiple classification and multiple inheritance. An instance may have no types, e.g. a custom-built component, or an instance does not have an assigned type yet.

By instantiating concepts defined on higher levels of abstraction, engineering experts can adapt or extend the plant knowledge-base according to their needs. This is only possible due to concept reification, which means that a given class can be regarded as the instance of a higher level meta-concept. This is the case for level M1 and M2 where everything is a class and an instance at the same time.

The advantage of using multiple modeling levels are, on the one hand, a clear separation of the modeling levels to integrate all aspects of the industrial plant provided by the domain experts. On the other hand, connections between the levels can be used to navigate between the levels, e.g. to navigate from a component instance “Motor m1” to its type, and to define constraints on usage and storage of the provided knowledge that can be validated during the plant engineering process.

8.1.3. Concepts of the Concept Model

A detailed specification of the main concepts of the concept model is given in the following. Such a specification is needed to define explicitly how the RMS has to process the knowledge incorporated in the partial plant models.

Entity

An entity is a concept that can have attributes and be connected to other entities by relationships. An Entity Instance can be connected to its Entity Type via the relation *entity type*. An Entity Instance can be an instance of several Entity Types, since multiple classification is permitted.

To illustrate the differences between the levels M0, M1 and M2, consider a *Hardware Component* as an example as shown in Figure 8.4. First, the author of a metamodel has to decide on Entity Metatypes, e.g. the author of the structure model defines three Entity Metatypes *Hardware Component*, *Hardware Component Template*, and *Hardware Component Instance*. An Entity Type like a Hardware Component Template can be instantiated by the manufacturer of a respective component, e.g. the manufacturer Siemens instantiates the concept Hardware Component Template for the component *Motor Siemens 1FK7*. An Entity Instance is instantiated by a plant operator, e.g. the *motor m1* is assigned a unique serial number.

Attribute

An Attribute may be an ***Attribute Class***, an ***Attribute Instance***, a ***Value Attribute*** or a ***Variable*** as presented in Figure 8.10.

An Attribute Class describes all characteristics of an Entity Type. An instance of an Attribute Class is an attribute of an Entity Instance and either an Attribute Instance, a Value Attribute or a Variable. Every Attribute Instance, Value Attribute or Variable has an *attribute type* that refers to an Attribute Class.

To show the usage of attributes, an example is given: Consider an Entity Type, e.g. *Motor*, which has an Attribute Class, e.g. *motor type* or *maximum torque* as shown in Figure 8.11. When the attribute behaves like an object, e.g. the Attribute Class *motor type = asynchronous motor*, then it corresponds to an Attribute Instance. In contrast to a Value Attribute, e.g. the Attribute Class *maximum torque* which has the Value Attribute *500Nm*

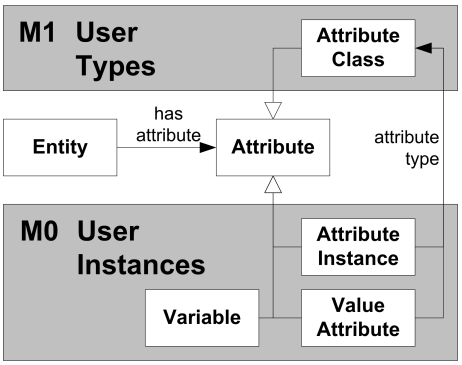


Figure 8.10.: Attributes in the concept model

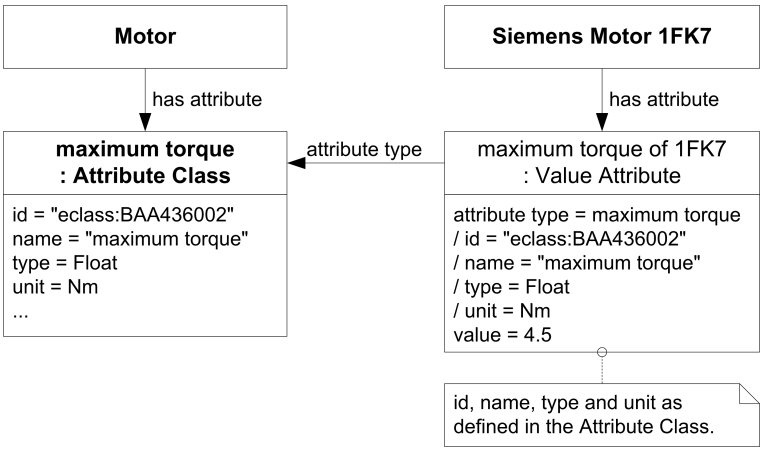


Figure 8.11.: Example: Attribute Class and Value Attribute

and corresponds to a Datatype, e.g. a string, a number or a quantity. An instantiation of the Motor is the Entity Instance *Siemens Motor 1FK7* which has the Value Attribute *4.5Nm* that refers to its Attribute Class *maximum torque*. Variables are attributes that are instantiated during operation of the plant (at run time). An example are attributes of motors such as "rotational speed" or "torque".

Three kinds of attributes were identified for Entity Types as depicted in Figure 8.12. The difference between these attributes occur during instantiation of a Hardware Component Template or during operation time. The attributes either get a value or change their values. A common example for entities are components which were chosen to demonstrate these four kinds of attributes:

- **type-specific attributes:**
  - Attribute Classes specified in an Entity Type, which are constant for all its instances, as shown in Figure 8.12. Examples for constant attributes are:
    - *company* = an attribute referring to the manufacturer of an Entity
    - *part number* = an attribute referring to the designation of a component defined by its manufacturer
  - Attribute Classes that have a default value in the Entity Type that may change in their instances, for example the Value Attribute of *max energy efficiency* defaults to *max energy efficiency ref*, but it can change when the operator measures the actual maximum energy efficiency in a realistic plant environment
- **instance-specific attributes**
  - Attribute Classes that have no default Value Attribute because they are different for each Entity Instance such as the attribute *serial number* which is only assigned by the manufacturer for Entity Instances

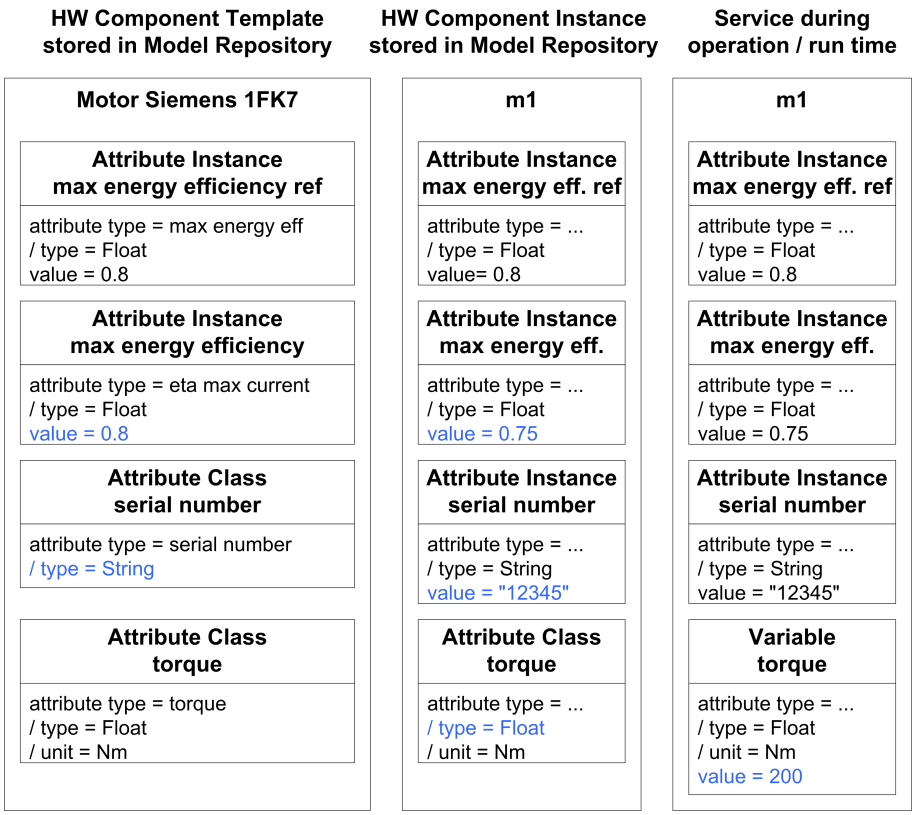


Figure 8.12.: Different kinds of attributes occurring in manufacturing plants

- **run-time attributes**
  - Attribute Classes that are only instantiated during operation, for example *torque* or *rotation speed* of a motor

Such a distinction is necessary to allow for an automatic monitoring of component attributes. The monitoring rules need to take into account what kind of attribute has to be monitored. For example, a rule for monitoring of type-specific attribute can be specified on level M1, by using the max or min values of the Entity Type attributes as threshold for all instances of this type. For instance a rule for the attribute “max energy efficiency ref” can be defined in a general manner, stating that “if max energy efficiency of any motor m1 of type “Motor 1FK7” is smaller than max energy efficiency ref then the motor m1 resides in an erroneous state”.

Identifier Every concept in a model must have an ID. The ID is assigned by using the attribute *id*. For example, the entity motor m1 has the attribute “id = m1”. IDs are unique for every concept. Every object within a plant can be accessed via its ID.

The identifier of a concept starts with its namespace, e.g. concepts of the metalevel have the namespace “meta”, entities in a conveyor plant have the namespace “conv”. IDs of concepts are extended with namespaces to group terms of similar contexts, e.g. the entity motor m1 of a conveyor unit has the ID “conv.m1”. Namespaces are also necessary to avoid name clashes between terms that have different meanings in different contexts. This is similar to namespaces in C#, OWL and packages in Java and UML.

Relationship

Similar to the concept Entity, Relationship Instance and Relationship Type are special cases of Relationship. A *relationship type* connects a Relationship Instance with its Relationship Type. A detailed model of relationships is presented in figure 8.13.

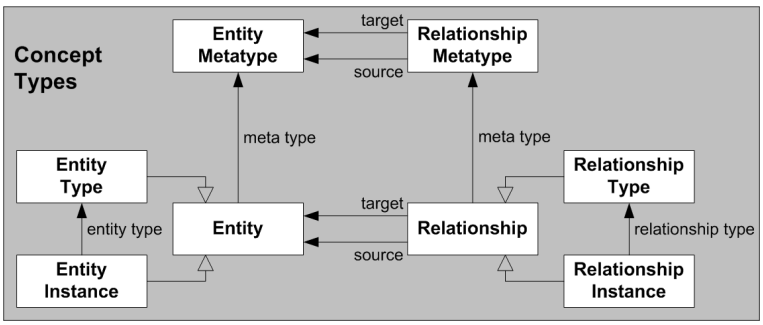


Figure 8.13.: Entities and relationships in the Concept Model

A formal definition of relationships in general is given in the following:

- a relationship $R \subset A\ x\ B$ is a subset of the Cartesian product A x B
- the ***domain*** of R: domain(R): = $a \in A | (a, b) \in R$
- the ***range*** of R: range(R): = $b \in B | (a, b) \in R$
- an n-ary relationship on a collection of sets $S_1, S_2, ...S_n$ is a subset $R \subset S_1 x S_2, ...x S_n$ of tuples of elements

To connect the different levels, relationships between the levels are required such as the relationship “entity type” that connects an Entity Instance with an Entity Type. Furthermore, relationships on every level of abstraction can be defined, e.g. the relationship “connected to” can be defined on level M2 to connect two components on level M0. Therefore, a relationship can behave in different ways, depending on the level where it starts and ends. The 5 possibilities to define relationships are shown in table 8.1.

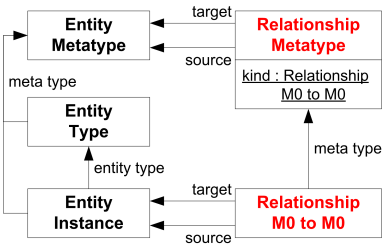
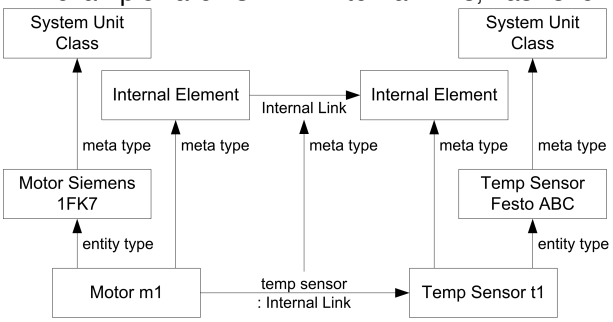
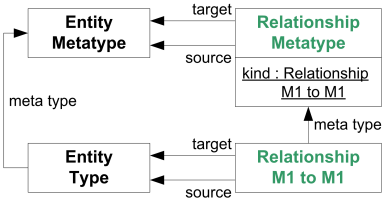
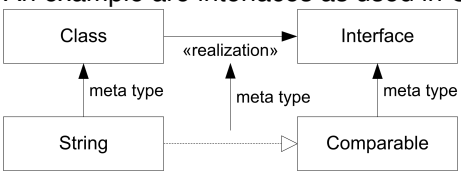
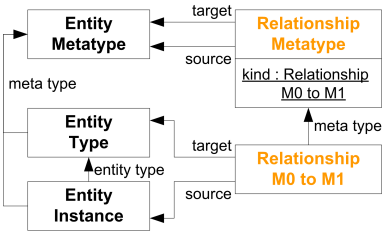
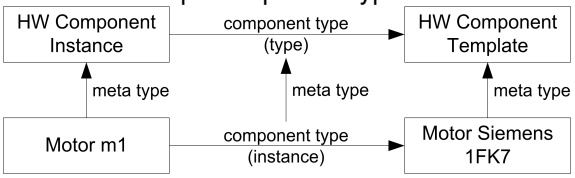
<table>
<tr><th>Level</th><th>Description</th><th>Diagram</th></tr>
<tr><td>M0 to M0</td><td>The simplest Relationship is a Relationship Instance connecting a User Instance (M0) to another User Instance (M0) without any requirements on User Types.</td><td>Relationship M0 to M0 in UML:
 
</td></tr>
<tr><td></td><td colspan="2">An example are CAEX InternalLinks, as shown in the following Figure:
 
</td></tr>
<tr><td>M1 to M1</td><td>There are also Relationship Types that exist only on M1 level. They connect a User Type (M1) to another User Type (M1) without any relationships between User Instances.</td><td>Relationship M1 to M1 in UML:
 
</td></tr>
<tr><td></td><td colspan="2">An example are interfaces as used in UML, C# and Java.
 
</td></tr>
<tr><td>M0 to M1</td><td>Relationships may connect different modeling levels. ***Instantiations*** are examples of Relationships that connect a User Instance (M0) to a User Type (M1).</td><td>Relationship M0 to M1 in UML:
 
</td></tr>
<tr><td></td><td colspan="2">Examples are all relationships ending with “type” in the concept model, e.g. the Relationship “component type” of the Structure Model:
 
</td></tr>
</table>

Table 8.1.: Different kinds of relationships in the concept model

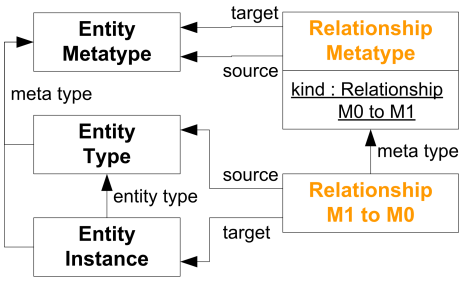
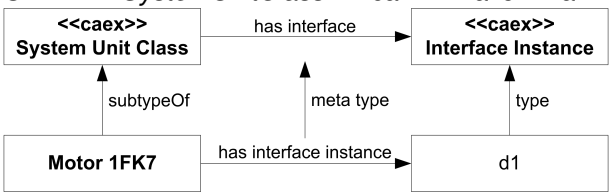
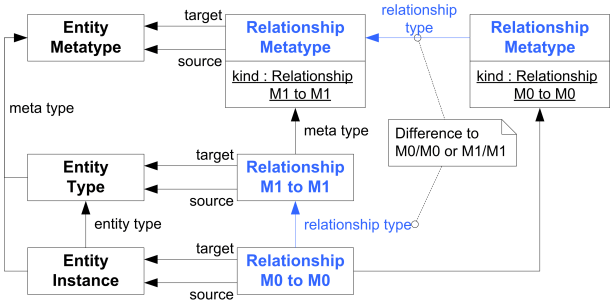
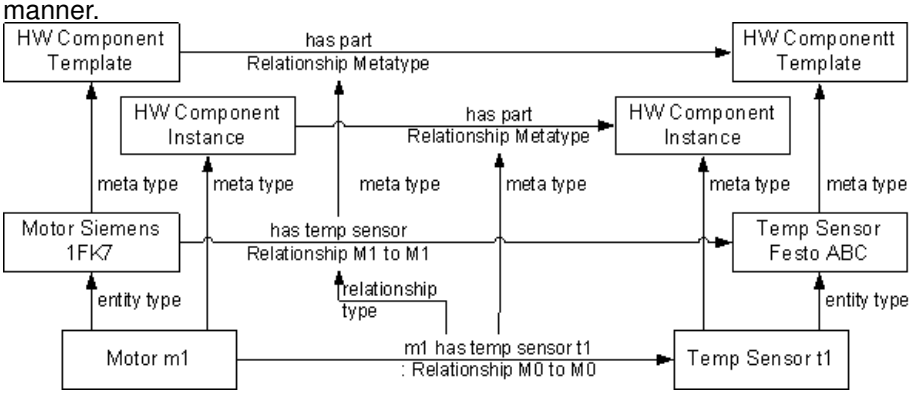
<table>
<tr><td rowspan="2">M1 to M0</td><td>There are also Relationships that connect a User Type (M1) to a User Instance (M0). That's the inverse relationship from the previous relationship.</td><td>Relationship M1 to M0 in UML:
 
</td></tr>
<tr><td colspan="2">An example fur such a Relationship exists in CAEX. A CAEX SystemUnitClass can have a CAEX ExternalInterface.
 
</td></tr>
<tr><td rowspan="2">M0 to M0 and M1 to M1</td><td>Most Relationship consist of a Relationship Type that connect a User Type (M1) to another User Type (M1), and several instances of the Relationship Type, that is, Relationship Instance that connects a User Instance (M0) to another User Instance (M0).</td><td>Relationship M0 to M0 and M1 to M1:
 
</td></tr>
<tr><td colspan="2">For example, the relationship has part of the Structure Model behaves in this manner.
 
</td></tr>
</table>

Table 8.2.: Different kinds of relationships in the concept model

8.1.4. Using the Concept Model

The previously defined concept model specifies basic concepts for several metamodels to provide an integrated view on the entire plant. In this section, several steps are proposed to specify how these concepts have to be used by experts of the engineering domain that want to build a knowledge-based RMS. To illustrate these steps, we use a very basic metamodel, the structure model which specifies structural facets of a plant such as the containment hierarchy and taxonomy of components, as an example.

- **Step 1:** Based on the concept model, engineering experts and knowledge engineers collaborate to construct metamodels that cover various aspects of an industrial plant. They define discipline-specific concepts as well as relationships to modeling concepts of related disciplines. The objective of the engineering experts, for example the author of the structure model, is to specify basic concepts that cover their discipline-specific expert knowledge needed for operating an industrial plant.

  Example: As shown in Figure 8.14, the author of the structure model defines Entity Metatypes and connects them with Relationship Metatypes. He defines the Entity Metatypes “Hardware Component Template (HC Template)” and “Hardware Component Instance (HC Instance)”. Both Entity Metatypes can be connected by the Relationship Metatype “has part”.

- **Step 2:** The previously constructed metamodels are used by the suppliers as a basis for modeling plant engineering data in form of user type models, which serve as templates for defining concrete instances. The aim of suppliers on the user type level is to describe characteristics of their components that can be verified and used by their customers for the monitoring purpose.

  Example: The suppliers can define instances of the HC Templates such as “Siemens Motor 1FK7” and of the Relationship Type “has part” which is detailed and named “has brake”. It becomes thus possible to verify on level M0 whether the instances have the correct “relationship type” as defined by the suppliers on level M1.

- **Step 3:** The previously introduced type models are instantiated in concrete plant models by the plant engineer. These instances represent the realizations of types as they are to be build up in reality. Instance models capture plant components, such as a specific motor with a certain serial number as it is assembled in the plant. The intention of the plant engineers on this level is to use the knowledge of the suppliers about the components for an appropriate handling and monitoring of the components.

  Example: The Entity Types of level M1 are instantiated and the Entity Instances “Motor m1” and “Brake b1” are connected with the more detailed Relationship Instance “m1 has brake b1”. This reflects the structure of a concrete plant as it is assembled from its components.

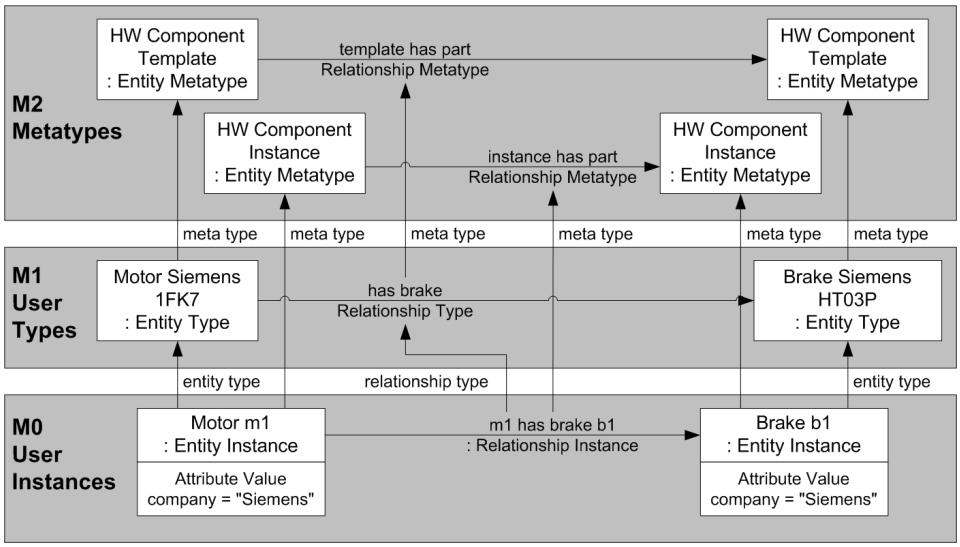


Figure 8.14.: Example for usage of the concept model for structural facets of an industrial plant

Strict adherence to these steps allow the domain experts to benefit from the following advantages:

- All partial plant models are derived from the concept model in a unified way and can be related with each other. This allows for an integrative view on an industrial plant across the respective engineering disciplines.
- The design decisions fixed in the concept model allow for an automated verification of the metamodels, the type models and the instance models with regard to their conformity with the concept model, e.g. by automatically highlighting any concept of the models that is not conform to concept model constraints. An example for such a restriction is the start and end point of relationships and their levels. These restrictions can be validated to guarantee a correct usage of the relationships by the suppliers or plant engineers, e.g. the relationship *part-of* can only be used to connect *HCs*, either on type or instance level.
- For any domain-specific aspect of an industrial plant, a custom metamodel can be introduced to later allow for domain-specific customized navigation and visualization of plant engineering data. This is a contrast to the predefined individual device descriptions that can be found in product catalogs of different manufacturers. The guided instantiation of the metamodels ensures a consistent view on the various aspects of an industrial plant by means of reusing model entities. For example, the instantiation of links with electrical wiring information, allows an engineer to navigate from a specific motor to its electrically connected power unit.

- The levels can be used to propagate knowledge from one level to another level. New facts can be generated automatically, e.g. the domain and range of relationships as defined in the metamodels are automatically propagated to level M1 and M0.

Another example for validations is shown in figure 8.14. In this example, the author of the structure model defines the concept Interface Class, which can then be used by the supplier. By using the structure model, the supplier can for example require interfaces for certain components. For instance, the manufacturer “Siemens” defines a *HW Template Motor Siemens 1FK7* which needs to have a *Relationship Type* “has driveCliq” to the interface “DriveCliq” as shown in figure 8.15. On the M0 level, the plant operator instantiates the *Relationship Type* “has driveCliq”. The resulting *Relationship Instance* “m1 has driveCliq d1” has to reside between an instance of the *Motor Siemens 1FK7* and an instance of *DriveCliq*, if this constraint is not respected, e.g. if the relationship “m1 has driveCliq m1” has correct start but wrong end point, an error occurs.

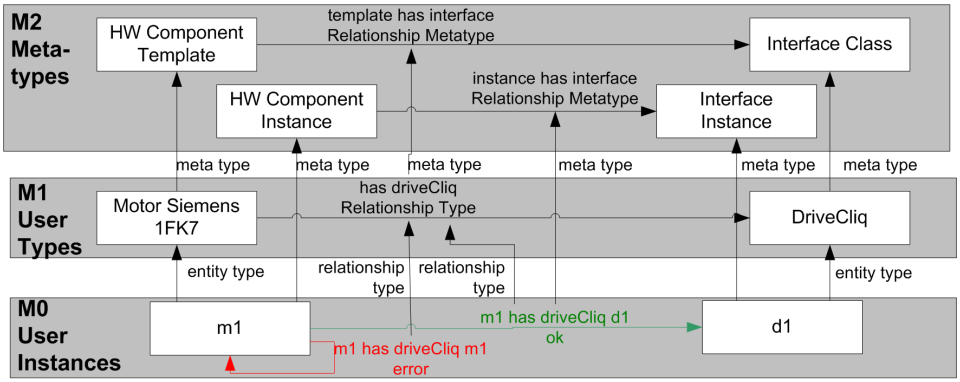


Figure 8.15.: Example for usage of the concept model for validation of relationships

Partial Plant Models needed for Resource Monitoring

By means of the conceptual modeling approach described in this section, all partial plant models can be integrated in the knowledge-base of the RMS so that they can be stored and processed in a common manner by the RMS. Within the RES-COM project, different responsibilities for partial plant models were assigned to the partners depending on the area of expertise of the experts that define the models, e.g. a control engineer defines the process model, a mechanical engineering defines the structure model. Overlaps between the models are allowed and occur when plant engineers have similar areas of expertise, e.g. mechanical engineers are usually concerned with both, structure and device info.

Several distinct partial plant models depending on each other were build based on the concepts of the concept model. An overview on the responsable process partners and dependencies between the models are depicted in Figure 8.16. A short description of all

partial plant models is given in the following, while the structure and the process model are defined in the subsequent sections of this thesis in more detail:

The **device model** describes the attributes of the hardware units in industrial plants. Optionally, this information can also be stored by the manufacturer on the component directly.

The **structure model** specifies plant taxonomy, plant topology and characteristic information about the plant and the devices (e.g. “the pneumatic cylinder has the role of a piston”). The structure model is discussed in Section 8.2.

The **process model** describes all activities included in the entire production process of an industrial plant. In addition, it relates the activities to components. A detailed description of the concepts of the process model is given in Section 8.3.

The **service model** describes all services that are executed by the components in a plant. This means that functions of the hardware components are regarded as services that represent the building blocks for the execution of the production process.

The **context model** includes context information, e.g. variation of energy costs during the day or utilization purposes of resources in a plant defined by manufacturing companies.

The **monitoring model** describes information to infer a state based on a monitoring condition by means of rules. The monitoring model uses all other models as a basis. A detailed description of concepts of the monitoring model is given in Section 10.2.

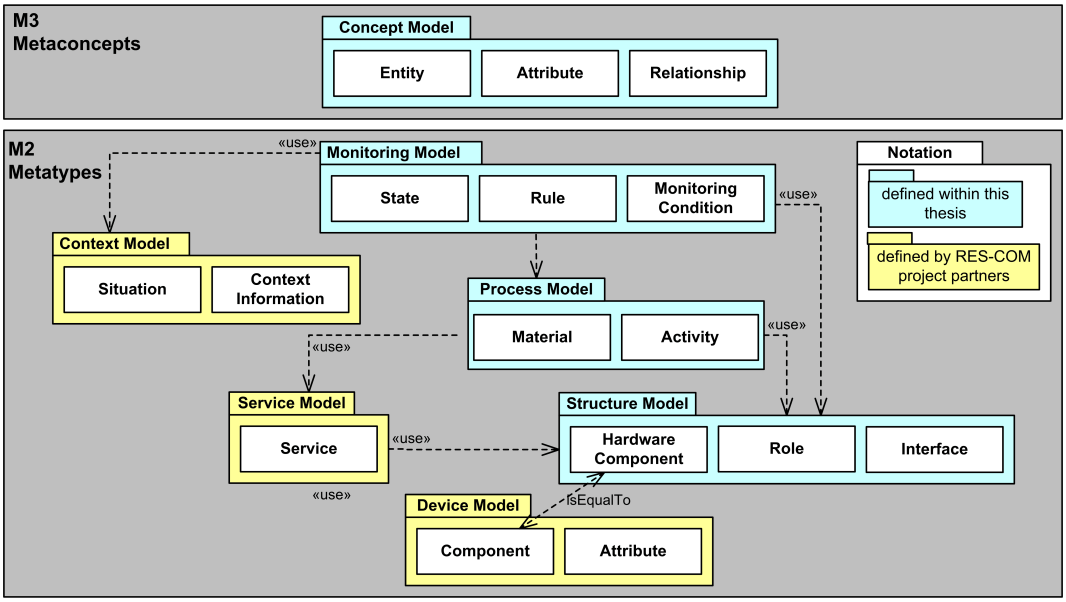


Figure 8.16.: Metamodels needed for resource monitoring and their dependencies

8.2. Structure Model

Modern facilities comprise a complex network of several objects with components and their subcomponents, bus systems, controllers and gateways. All these parts are usually handled separately and the structural information about them is stored in different tools (e.g. COMOS [123], Eplan [68], CAD tools [113]). But the information in modern plants is growing, because the complexity of control systems and the amount of sensors installed in the plants is increasing. To store all this information in a common place, a generic structure model to store structural facets of the plant becomes necessary.

The structure model is used to define three kinds of fundamental information: 1) the ***component taxonomy*** specifies and names element classes and arranges them into a classification hierarchy, 2) the ***plant topology*** defines the containment hierarchy of plant components and other functional relations between components such as *connected to* or *flow*, 3) ***characteristics*** about the plant and the components, e.g. the role or the attributes of a component.

Based on our experiences with related standards, mainly CAEX, and experts of the industrial domain, some basic entities for the structure model as listed in Table 8.3 were identified. These entities are needed to allow plant engineers to model the structure of an industrial plant, but the structure model can be extended by further entities if necessary.

Entity	Entity Type	Entity Instance	Definition
Element	***Element Type***	***Element Instance***	a fundamental concept that can be used to extend the structure model
Hardware Component	***Hardware Component Template***	***Hardware Component Instance***	a basic, self-contained entity of an industrial plant. Refers to a manufactured component that can part of a larger component.
Interface	***Interface Type***	***Interface Instance***	a concept that refers to a point of interaction between Hardware Components.
Port	***Port Class***	***Port Instance***	used to describe composite interfaces that consist of multiple single interfaces
	Role Class		describes an abstract functionality of a component without defining the underlying technical implementation

Table 8.3.: Concepts of the structure model

A UML diagram in Figure 8.17 presents the relations between basic entities of the structure model. Due to the complexity of the structure model, only the main relationships can be represented in this diagram. A more detailed list of the relationships introduced in the

structure model according to the modeling constructs of the concept model is shown in Figure 8.18:

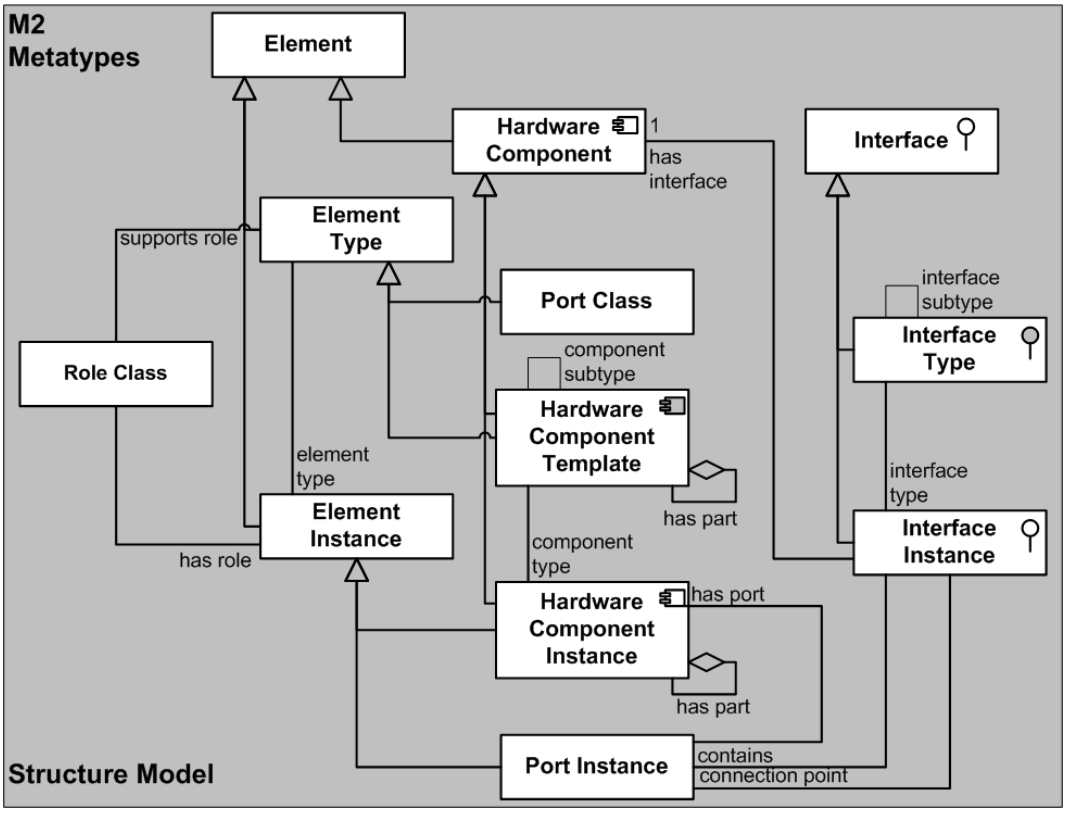


Figure 8.17.: UML class diagram including all entities of the structure model and their relationships

8.2.1. Concepts of the Structure Model

This section embodies a more detailed description of the entities defined in the structure model. Such a description is needed to specify the usage of the concepts.

Element

An Element is an advanced concept that can be used to extend the structure model with additional concepts. An example for such an extension is the definition of a new concept “Group”. This new concept can simply be added as a subconcept of the concept Element. The according relationships to other concepts are then still valid and no additional concepts or relationships need to be added.

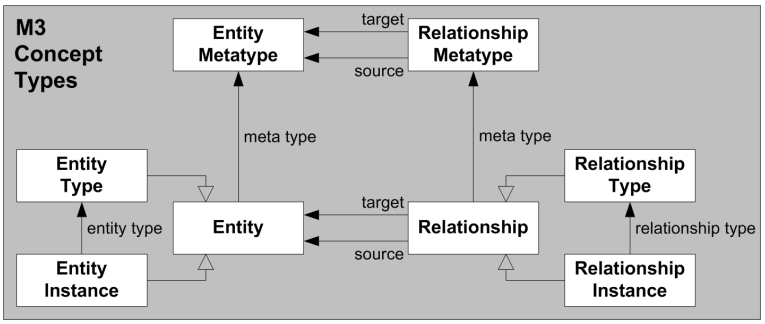


Figure 8.18.: Relationships introduced in the structure model

An Element is either an Element Instance or an Element Type. There is no distinction between those elements that have a physical existence and those that have not. Both kinds of objects can be relevant for being identified and handled in the life-cycle of a system. An Element Instance uses the relationship *element type* to refer to its Element Type.

Hardware Component

Description: A Hardware Component describes a component which is part of either an industrial plant or product. A hardware (HW) component can be composed of other HW components. In this case the component is a *composite component*.

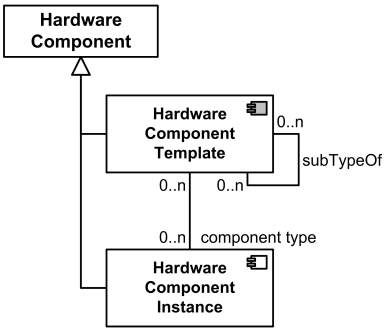


Figure 8.19.: UML diagram of HW Components

A HW Component is either a HW Component Template or a HW Component Instance:

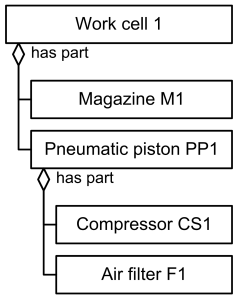
- A HW Component Instance describes a concrete *physical* hardware component in an industrial plant or product.
- A HW Component Template describes an abstract *type* or *specification* of a hardware component as specified in manufacturer's product catalog.

The relationship *component type* connects a HW Component Instance with its HW Component Template. A *HW Component Instance* can have several *HW Component Templates* and a Template can have several *supertypes* (expressed by its inverse relationship subTypeOf). The detailed UML diagram that specifies a HW Component is presented in Figure 8.19.

The origin of the notion of hardware component is reflected by a generalization of terms from the following sources:

- English terms defined in the standard IEC 62264 [4]: Component, Group, Work Cell, Production Line, Area, Site
- German terms defined in the German standard DIN 40150[5]: Baugruppe, Baueinrichtung, Bausystem

Example of Usage: Composition of HW components is expressed with the relationship *has part* as shown in the example in the Figure on the right. The composite components here are the “Work cell 1” and the “Pneumatic piston PP1”. The Structure Model does not impose fixed rules on when a set of parts should be modeled as a component. It is the designer/engineer who decides that constitutes a component. When a component is established, information may be associated with it. This information may change throughout the life cycle of the component.



Interface

Description: To define connections between components, in some cases not only relationships are required, but also a reference to a point of interaction between components. This point of interaction is called *Interface*. Interfaces can be linked with each other and can be implemented on hardware or software level. A component can have multiple interfaces. An individual connection point is an ***Interface Instance***, e.g. usb port 1 and usb port 2; and has an ***Interface Type***, e.g. USB 2.0 Port. A Hardware Component Template has Interface Instances, not Interface Types. A Hardware Component Instance has the same Interface Instances as its Hardware Component Template and it can have several Interface Instances of the same Interface Type as shown in Figure 8.20.

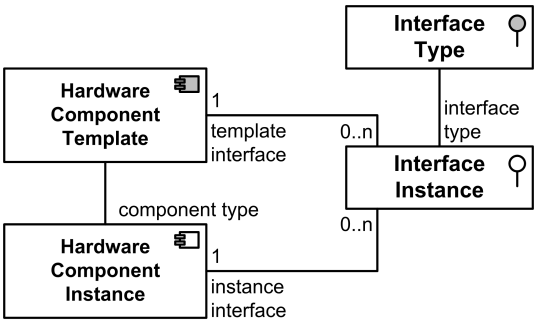


Figure 8.20.: UML class diagram of Hardware Components and Interfaces

A Relationship between HW Components, e.g. a pressure valve Y1 and a pipe PS1, can either be defined directly or via Interface Instances, e.g. Pneumatic in or out, attached to the components as presented in the example in Figure 8.21.

Therefore, the concept ***Relationship Endpoint*** is introduced. The Relationship Endpoint is a common supertype of Hardware Component and Interface Instance and used to connect source and target of a relationship as depicted in Figure 8.22.

[4] IEC 62264-1 (2003): ” Enterprise-Control System Integration Part 1: Models and Terminology”
[5] DIN40150 (1979): “Concepts for the arrangement in connection with functional and constructional units”

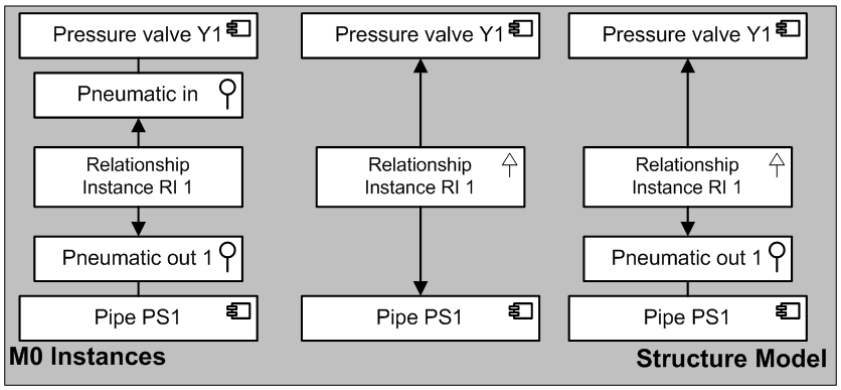


Figure 8.21.: Example for different kinds of connections between HW components

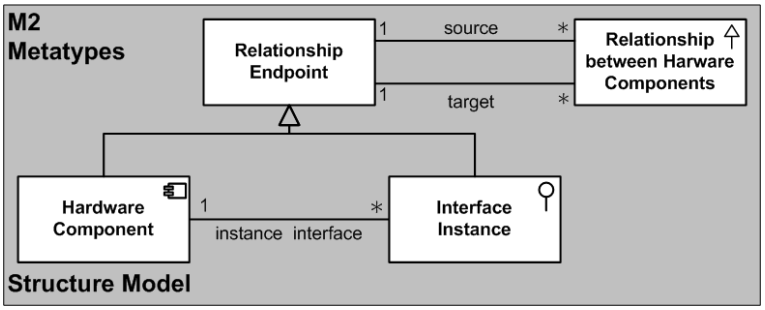


Figure 8.22.: UML class diagram of the concept *Relationship Endpoint*

Example of Usage: Examples for such an interface are *Signal Interfaces* which have the attribute "*direction = input / output*" to indicate the data flow and an attribute "*type = digital/ analog*". Such a specification allows for an automatic validation mechanism to check whether the relationship is correctly implemented, i.e. an input is connected to an output or a digital interface is connected to an interface of the same type. A more detailed implementation of this verification example is presented in more detail in section 12.2.3. Figure 8.23 shows a component Compressor CS 1 has an interface Pneumatic out 1 which is connected with the relationship connected to with the interface Pneumatic in of the component Pipe PS1. Pipe PS1 is connected similarity to Pressure valve Y1.

Port

Description: A Port is used to model complex Interfaces. In AutomationML, the port concept is used to describe composite interfaces that consist of multiple single interfaces. A Port is either a Port Instance or a Port Class. AutomationML, SysML and UML define Port with similar descriptions.
Example of Usage: A Hardware Component Instance might have a Port Instance. A Port

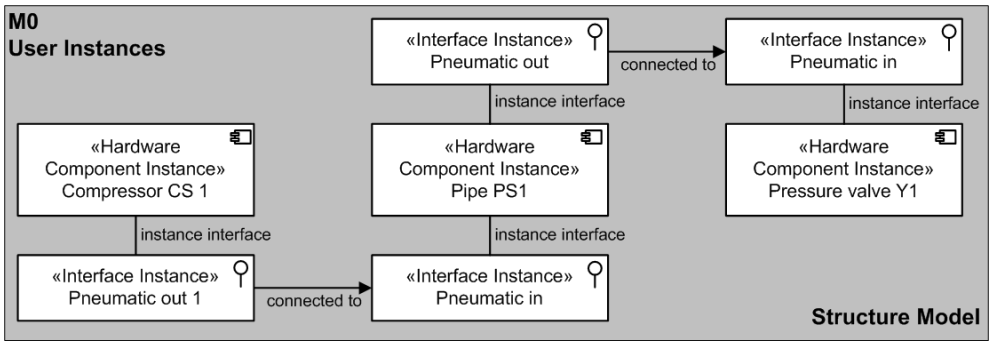


Figure 8.23.: Usage of Interfaces for realizing a pressure system

Instance contains one or multiple Interface Instances. These Interface Instance can all be connected with another Port Instance via a connection point of the ports as presented in Figure 8.24.

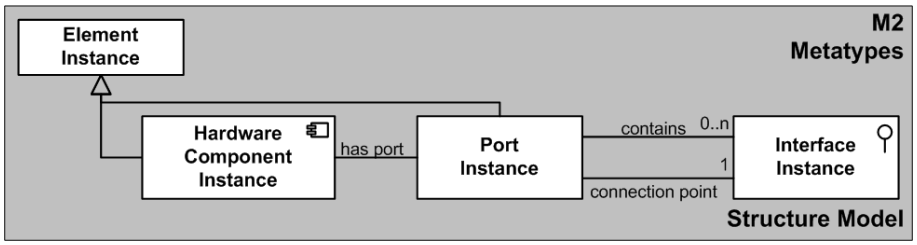


Figure 8.24.: UML Class diagram of a Port Instance and relations to other concepts

Role Class

Description: A *Role Class* is a class that describes an abstract functionality without defining the underlying technical implementation. Role class examples are a *Robot*, a *PLC* or a *Conveyor* as specified in CAEX. Several applications were identified for roles in an industrial plant.
Example of Usage: Role Classes have multiple purposes, the can be used to:

1. Choose appropriate hardware components during the *plant engineering process*. An example for this task is shown in Figure 8.25: a conveyor is required to transport a washing machine, thus we need a role ”washing machine conveyor” with specific requirements. The information about HW components specified in the product catalog of a manufacturer can then be used to identify the HW Component Templates that fulfill the requirements defined in the role.

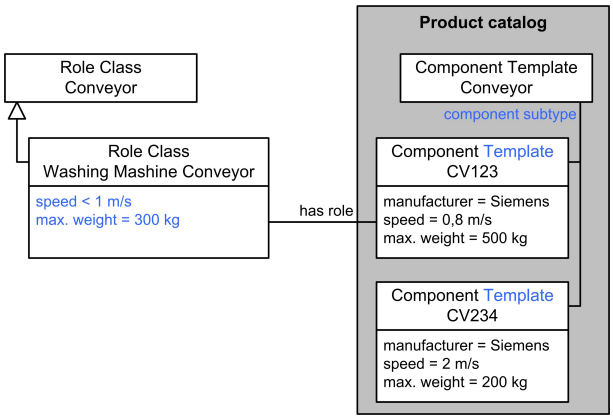


Figure 8.25.: Selection of Hardware Component Templates that fulfill the requirements defined in the Role Class “Washing Machine Conveyor”

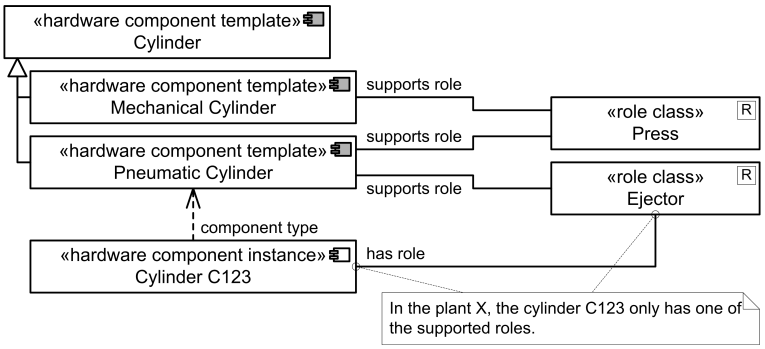


Figure 8.26.: Example for a HW Component Template supporting multiple roles

2. Define several functions for one component or a function that can be fulfilled by several components, e.g. a pneumatic cylinder can be used as an ejector or a press, depending on his specific task in the plant as shown in figure 8.26. Several components can support the same role, e.g. the role *press* can be supported by a pneumatic or an electric cylinder

3. Distinguish between process-product-resource as proposed by AutomationML. Roles are important for constructing and maintaining a plant. The plant engineer can thus change the components in the manufacturing chain depending on the context, e.g. if the product has to be produced in a resource efficient manner, then better use an electric cylinder than a pneumatic cylinder (which has a bad CO2-footprint) as a press.

8.2.2. Plant Taxonomy

The **plant taxonomy** is required to specify all entities in a plant, and arrange them into a classification system. Most modeling languages for software and CAE (Computer Aided Engineering) use types for this purpose. The main benefits of the usage of types are that: 1) Typed objects provide type safety and clear semantics of attributes and relations, 2) type-specific attributes can be used to reduce redundancy and increase reusability.

Three ways of modeling were identified to design a taxonomy of devices in the model repository:

- usage of ***types***
  - a classification hierarchy can be established with the *subtype of* relationship
  - such a hierarchy corresponds to the product catalog of the manufacturer
- usage of ***attribute classes***
  - a taxonomy of attributes is defined. This corresponds to a taxonomy of components, e.g. the ***attribute class*** *motor type* has several ***attribute instances*** such as “asynchronous motor” or “synchronous motor” with according subtypes “AC motor”, arranged in a taxonomy
- usage of ***role classes***
  - a taxonomy of *role classes* is defined and specific roles are assigned to the components, e.g. a motor “Siemens Motor 1FK7032-5AK71” has the role class “conveyor drive” which is a subtype of the role class ”drive”

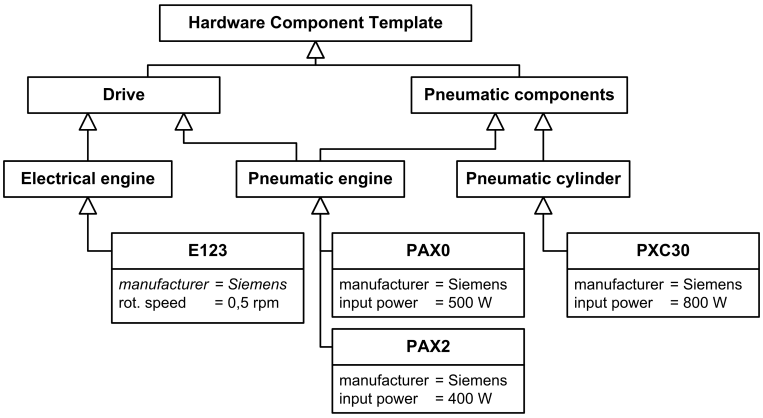


Figure 8.27.: Exemplary component taxonomy of an industrial plant

The *HC Templates* may be arranged into a library of components which is provided by the manufacturer of the plant components similar to a product catalog. Different kinds of

hierarchical classifications are possible to structure such a library as shown in Figure 8.27: *functional groups* classify components with identical functions, e.g. a HW component template “Drive” groups drives including “electrical engine” and “pneumatic engine”; *material groups* classify components which use same material as input, e.g. the HW component template “Pneumatic components” has the subtypes “pneumatic cylinder” or “”pneumatic engine”.

8.2.3. Plant Topology

A Plant Topology[6] encompasses not only the containment hierarchy or partonomy of plant components as defined by the *part of* relation, but also other functional relations between components such as *connected to* or *flow*.

A component in an industrial plant may contain other components. The resulting component containment hierarchy is also called **partonomy**. Every plant has a different partonomy and therefore specific amount of hierarchical levels of components. Mechanical engineers describe such hierarchical levels by encapsulating the components in production lines and workcells as shown in figure 8.28.

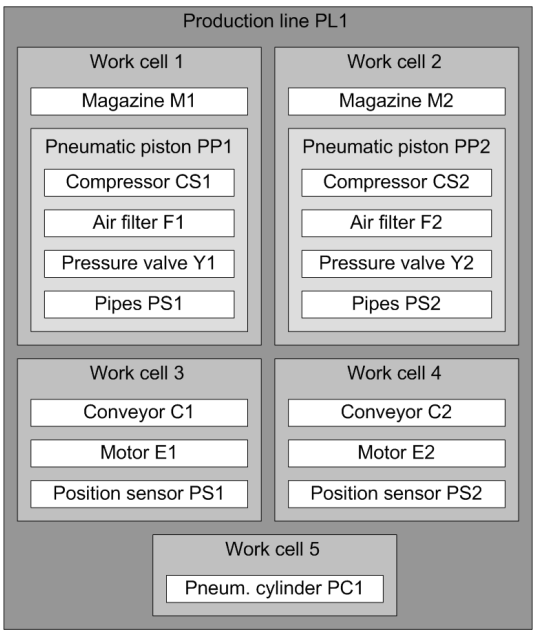


Figure 8.28.: Example for encapsulation of plant components in a production line

Companies usually have their own terms and granularity for the hierarchical levels of a plant depending on the constraints imposed by their regulations and Key Performance

[6] The term topology is used here in the sense of a physical structure of a network as specified in [73]

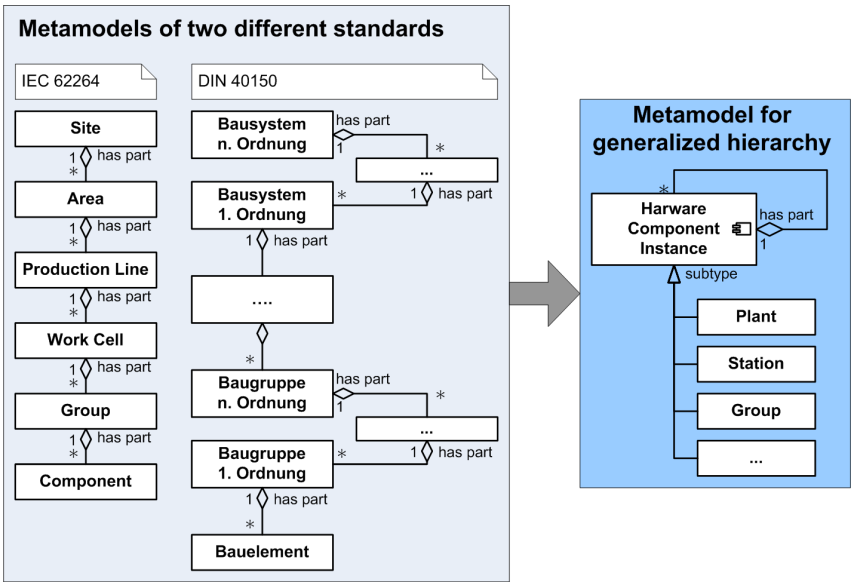


Figure 8.29.: Hierarchical levels defined in standards and generalized plant hierarchy

Indicators (KPIs). The hierarchy of standard IEC62264-1[7] defines exactly six levels of encapsulated components for a partonomy. This is in contrast to the hierarchy of standard DIN40150[8], shown in Figure 8.29, which allows for defining an unlimited amount of encapsulated components. Thus, the challenge in defining a topological structure is to find an extensible generalized hierarchy which matches all plant hierarchies.

In order to generalize the hierarchy, the concept of *Hardware Component* of the structure model can be used as generic term for area, production line, work cell, etc. Additionally, the relationship *has part* is used to express that one component is part of another component. The relationship *has part* is transitive and its inverse relationship is *part of* such that for a set of components C:

$\forall x, y, z \in C : x \text{ has part } y \wedge y \text{ has part } z \Rightarrow x \text{ has part } z$

$\text{part of}^{-1} = \{(y, x) \in Y \times X \mid (x, y) \in \text{has part}\}$.

The result as shown in Figure 8.29 is a simple generalized hierarchy which can express all kinds of hierarchical levels by usage of subtypes. The inverse relationship *part of* allows to navigate from lower component levels to the upper levels of the hierarchy and vice versa.

Furthermore, the structure of an industrial plant is heavily depending on the mechanical or electrical connections between its HW Components. A relationship *connected to* is thus required to describes physical or electrical connections between HW Components. Subproperties such as *electricalConnectionTo* or *mechanicalConnectionTo* are defined to

[7] IEC 62264-1 (2003): " Enterprise-Control System Integration Part 1: Models and Terminology"

[8] DIN40150 (1979): "Concepts for the arrangement in connection with functional and constructional units"

specify the connection in a more precisely. The relationship *connected to* is transitive and symmetric so that:

$\forall x, y, z \in C :$ *x connected to y* $\wedge$ *y connected to z*
$\Rightarrow$ *x connected to z*,

$\forall a, b \in C :$ *a connected to b* $\Rightarrow$ *b connected to a*.

An example for its usage as presented in Figure 8.30 allows to state that pressure valve *Y1* is connected to pipe *PS1* which is connected to a compressor *CS1*.

Plant engineering experts can extend the structure model together with the knowledge engineer in order to define their own plant-specific relationships. The task of the knowledge engineer is to ensure the formal properties of the model in alignment to constraints defined in the concept model. An example for such a relationship is the *heatFlow* relationship which is needed to state that a Hardware Component transmits heat to another Hardware Component, e.g. a boiler *b1* transmits heat to a motor *m1*. This relationship is transitive and symmetric, thus:

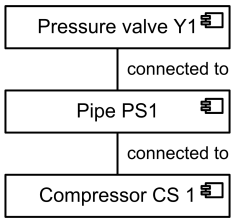


Figure 8.30.: Usage of relationship "connected to"

$\forall x, y, z \in C,$ *x heatFlow y* $\wedge$ *y heatFlow z* $\Rightarrow$ *x heatFlow z*,

$\forall a, b \in C,$ *a heatFlow b* $\Rightarrow$ *b heatFlow a*.

8.2.4. Plant Characteristics

The last element of the modeling concepts is the description of the plant characteristics. The individual characteristics of components in a plant are represented in form of attributes. A distinction is made between attribute classes, e.g. every component has an Attribute Class *manufacturer*, and Value Attributes e.g. the Value Attribute of *motor m1* is *Siemens*.

During the construction process of a plant, the plant engineer selects and specifies the components in a plant step by step. The stages he executes are listed in the following to ensure that all concepts needed during this process are covered by the structure model.

The construction of an industrial plant runs through several stages. During the first stage, the plant engineer defines abstract *Hardware Components* to build the plant hierarchy including connections between the components. This includes all components needed for executing the process steps required to build the final end product. In a second step, he assigns specific roles for every abstract plant component. Such a role is for instance an ejector which is used to push parts from a magazine onto a conveyor or a press that pushes two parts together. In a third stage, the engineer has to refine the abstract roles by assigning requirements to the roles that have to be respected. For example, a conveyor has an attribute maximum speed of 1 m/s and can transport pieces with maximum weight of 300 kg. Based on the role requirements, he selects in a fourth stage the specific plant

components of a product catalog provided by the component manufacturer's. The characteristics of the products in the catalog are checked, e.g. the speed limit of a conveyor and the maximum supported weight, see Figure 8.25.

In a last stage all selected components are mounted to an entire facility. Currently, different tools and methods depending on the industrial area and the plant environment are used during these stages. However, the purpose of the structure model is to insert all information specified during the entire plant construction process in one model.

8.3. Process Model

To determine the monitoring state of components, the current process of the component or remote components has to be considered. The required process information can be divided in two parts: a static model of the production process and the dynamic information about the current process state. Today, this information is typically contained implicitly in the control procedures of the process control devices like programmable logic controllers (PLCs). Since the control procedures of these controllers are generated on a low implementation level, where the individual binary in- and outputs are processed, the code is complex and monolithic. Therefore, this procedure should be developed on a higher abstraction level based on the concept model.

The process model used for the RMS builds on ontology patterns of a wide-spread ontology source, the Process Specification Language (PSL) which "identifies, formally defines, and structures the semantic concepts intrinsic to the capture and exchange of discrete manufacturing process information" [117] and some additional entities and relations to other partial plant models. The following PSL ontologies are used for the RMS:

- PSL Core
- Subactivity Extension
- Activity-Occurrences Extension

But the process model is not limited to activities. It further specifies the material that is processed by the activities and manipulated by the components during the entire production process. The main entities introduced in the process model are presented in Table 8.4:

Entity	Entity Type	Entity Instance	Definition
Activity	***Activity Type***	***Activity Occurrence***	A class of actions as defined by the PSL ontology. Activities are part of the entire production process.
Material	***Material Type***	***Material Occurrence***	Material is *manipulated by* components in a plant and *processed* by activities.

Table 8.4.: Concepts of the process model

8.3.1. Concepts of the Process Model

This subsection embodies a more detailed description of the entities defined in the process model. Such a description is needed to specify the usage of the concepts. The detailed process model is depicted in Figure 8.31. In contrast to the structure model, the process

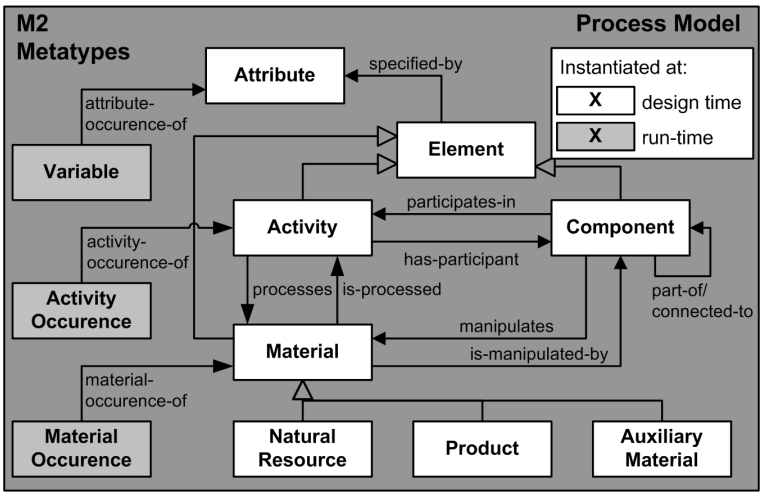


Figure 8.31.: UML class diagram of the process model and its relations to the structure model

model has to cover also dynamic information about the current process state. Therefore, a distinction between entities instantiated at run-time and at design-time is made.

Activity

Activities are a type of action. Activities are executed by HW Components in industrial plants. Every HW Component can execute one or more activities. ***Activity Types*** are considered as to be reusable behaviors within the domain. Activities may have multiple occurrences, or there may exist activities that do not occur at all. For example, 'movePart' is an activity. It is the class of actions in which parts are being moved. An example for a subtype of the Activity 'movePart' is the Activity Type 'movePartFast' is the class of actions in which a motor moves an conveyor quickly and "movePartFast executed by m1 at 2 PM on May 25, 2013" is an occurrence of the activity 'movePart'.

An ***Activity Occurrence*** is associated with at least one unique Activity Type by the relationship *activity-occurrence-of* and begins and ends at a specific point in time. Multiple relationships were defined in PSL to denote an ordering over activity occurrences, so that they form trees. An entire tree represents the structure of process sequences of an industrial plant. A more detailed description of these concepts and especially the relationships between activities, such as *before* is given in [117].

Material

Material encompasses raw material, natural resources and products required for value generation in an industrial plant. The material is manipulated by the HW Components and processed by Activities. Material is also the supertype of all entities needed for producing

an intermediate or final product in an industrial plant. These objects are either consumed within a production process or enter a production process. Main examples for subtypes of ***Material Types*** are ***Natural Resource***, ***Product*** or ***Auxiliary Material***. An example for a natural resource is 'Energy' or 'Water'.

A ***Material Occurrence*** is associated with at least one unique Material Type by the relationship 'material-occurrence-of'. An example for a subtype of the Material Type "Product" is 'SteelEndproduct' used to describe the class of end product produced by steel manufacturing plants and its instantiation 'steelEndproduct1 produced by a Thyssen plant1 at 2 PM on May 25, 2013'.

9. Evaluation of Application-Independence

A conceptual modeling approach as theoretical basis for integrating knowledge on different plant aspects in the RMS was presented in the previous chapter. To evaluate the concepts of the conceptual modeling approach, the second research question is applied:

> **Research Question 2:** How can knowledge-based techniques integrate knowledge on different plant aspects to realize a resource monitoring system which is universally applicable independent of the branch of industry and the manufacturing engineering area?

The degree of reuse is a relevant factor to evaluate the application independence of the conceptual modeling approach. The approach was thus used in two different reference projects while tracking the degree of reuse. The following evaluation question was defined for this purpose:

> **Evaluation Question 2:** Is the approach applicable to industrial plants of either distinct branches of industry or manufacturing engineering areas?

The conceptual modeling approach and the partial plant models described in Chapter 8 have been originally defined in the context of the RES-COM project with a focus on discrete manufacturing. The resulting approach has been further used in a second research project I2MSteel in the steel manufacturing industry. The I2MSteel project[1] aims towards the development of an agent platform to facilitate the seamless high-level information exchange across the supply chain by means of semantic technologies.

To evaluate if the approach is also applicable for use cases in the steel manufacturing industry, workshops were conducted with knowledge engineering experts of the I2MSteel project. Within this workshops, we taught them how to specify partial plant models by means of the conceptual modeling approach. It became obvious that most of the concepts of the concept model and main partial plant models defined within the RES-COM project could be reused in the I2Msteel project. To determine the degree of reuse, concepts of all models used in both projects were compared. Then, the number of similar concepts (with similar semantic meaning) of models were counted and divided by the total number of concepts specified in the model, such that for each model m the degree of reuse is:

[1] more details on the I2MSteel project can be found at `ftp://ftp.cordis.europa.eu/pub/coal-steel-rtd/docs/summaries-rfcs_en.pdf`

$$\textit{degree of reuse}(m) = \frac{\textit{number of similar concepts}(m)}{\textit{total number of concepts}(m)} \tag{9.1}$$

The core models of RES-COM are the concept model, the structure model, the process model and the monitoring model. For the core model on level M3 which specifies basic concepts for all other models is represented by the concept model. A degree of reuse of 100% was determined for the concept model in I2MSteel. A detailed discussion on this finding was also published in [162]. The other partial plant models had to be adapted partially to be suitable for the use cases of the steel production domain as described in the following.

The structure of plants in the discrete manufacturing and steel manufacturing is very similar. Therefore, it was not surprising that concepts of the structure model were reused to 80%. Only the concepts `Group Class`, `Port Class` and `Port Instance` were not instantiated for I2MSteel use cases. The Process Model was divided into a Process Model and a Product Model, since the product-related information is a crucial factor in steel manufacturing. The Process Model relies in both projects on the PSL standard. But in the I2MSteel project, a detailed distinction is made between the product types “liquid steel”, “slab” and “coil” inspired by the Production definition model of ISA-95[2]. The degree of reuse of the Process Model adds up to 80% including the Process and Product Model. The Monitoring Model is named Measurement Model in the I2MSteel project and reused to 80%. The purpose of the Measurement Model was to specify a more general term to use it for different applications, such as monitoring or diagnostics, by capturing information on relevant measurements of the steel production processes. An overview on the degree of reuse of all partial plant models within the I2MSteel project is depicted in Table 9.1.

RES-COM	**I2MSteel**	**Degree of reuse**
Concept Model	Concept Model	100%
Structure Model	Structure Model	80%
Process Model	Process Model	70%
	Product Model	10%
Monitoring Model	Measurement Model	80%

Table 9.1.: Degree of reuse of partial plant models of research project RES-COM in I2MSteel

An average degree of reuse of 85% could be determined within the workshops in the I2MSteel project. This high degree of reuse clearly proved that the described conceptual modeling approach can be applied to different use cases in the manufacturing domain, where knowledge of different engineering experts has to be integrated and analyzed. Although the RES-COM and the I2MSteel project focus on different manufacturing engineering areas, the applications and use cases of these projects can both be addressed by means of the concept model and the partial plant models specified in this thesis.

[2] ISA-95: “Enterprise-Control System Integration”, see http://www.isa-95.com/

10. Modeling Approach for Resource Monitoring

A resource monitoring system observes the individual resource usage of plant elements to analyze their resource usage while considering context information such as the current process step or varying energy costs. The assessment of the resource usage and the usage efficiency is a complex task which requires additional knowledge about resources used by the plant. A systematic approach for the knowledge-based integration of relevant information is thus presented in this chapter. Such an approach is a key requirement to infer reliable monitoring states of plant elements.

Current solutions for monitoring systems in industrial production face several issues. As stated in Section 2.1.1, monitoring systems for industrial plant should not consider only sensor data, but various input parameters, such as the current process step or the state of remote components, to compute the actual monitoring state of a plant element. Additional challenges are the flexible manufacturing environment and the high implementation costs as pointed out in Chapter 4. This is the motivation for investigating the use of knowledge-based models and inference rules for realizing an RMS. The rules can be used to represent monitoring knowledge in a declarative yet executable way, while plant models support the representation and organization of plant knowledge used as a basis for formulating monitoring rules.

Another issue concerning rule-based monitoring approaches is that several domain experts are typically involved in the engineering phase of monitoring systems and generate a huge amount of monitoring rules. Difficulties arise, however, when contradicting or redundant rules are generated and when the number of rules exceeds a certain limit that the experts cannot cope with. In order to support these experts, ontologies can be used as an umbrella for organizing and structuring monitoring rule constructs, next to the representation of plant knowledge, to support the engineering and maintenance of the monitoring rules.

The goal of this chapter is to generate a systematic approach for integrating plant knowledge and rules in the monitoring system by offering user support mechanisms and a high degree of reconfigurability. To this end, a decentralized monitoring architecture is specified for the RMS in Section 10.1. An umbrella model for the purpose of monitoring is presented in Section 10.2. This model is used to relate all models required for monitoring and for structuring monitoring rule constructs. These rule constructs need to be maintained as explained in Section 10.3, to ensure an effective monitoring on a long-term perspective.

10.1. Decentralized Monitoring Approach

As discussed in the previous section, the assessment and analysis of resource usage in the manufacturing environment is a complex task which requires additional knowledge and input data of the plant. Nowadays, monitoring systems run on a central unit that collects sensor data of the plant to compute the monitoring states of the different components. But RMS need to make the plant engineers aware of the resource usage of their plants on the level of the incorporated components as stated in [4]. Modern plants require furthermore a higher degree of flexibility and thus a decentralized system architecture where components are able to monitor themselves by means of intelligent autonomous devices such as active digital product memories (ADPMs) as specified in [122].

Such a decentralized architecture of the RMS allows to equip components on all plant levels with monitoring functionality by placing monitoring nodes at various positions in a plant's installation. The advantage of such a decentralized system with autonomous devices are manifold: (1) the manufacturer's produce intelligent components that can monitor themselves, using their extensive knowledge about the products, (2) an exchange of single components does not require modification of the entire system and downtime, (3) failure of one unit of the RMS will not affect the operation of the entire RMS.

The main modules of the decentralized RMS proposed within this thesis as shown in Figure 10.1 are:

1. Every component in the plant is equipped with an ADPM including a monitoring unit (MU). An ADPM consists of a knowledge base that contains knowledge about the plant in a machine processable way, including a rule base. For our monitoring purpose, we distinguish between three kinds of input, 1) structural information about the plant, e.g. motor is monitored by a temperature sensor and a smart meter, 2) current process step, e.g. conveyor that is driven by the motor is executing the activity "transport at maximum speed", 3) context information of the plant, e.g. plant is producing at half load. This input and the knowledge-based models and rules are then processed by an inference engine in the monitoring unit to compute the monitoring state.

2. Collections of components are combined to groups. Every group has its own decentralized MU which takes the individual component states as input to compute a composite state of the group.

3. The heart of the system is the monitoring unit of the industrial plant. It uses component and group states as input to compute the state of the entire plant. A knowledge base editor assists the plant engineers in their engineering task and performs consistency checks on the knowledge base.

4. The plant monitoring state computed by the MU of the plant are presented to the plant engineer. Based on the resulting states provided by the monitoring system the

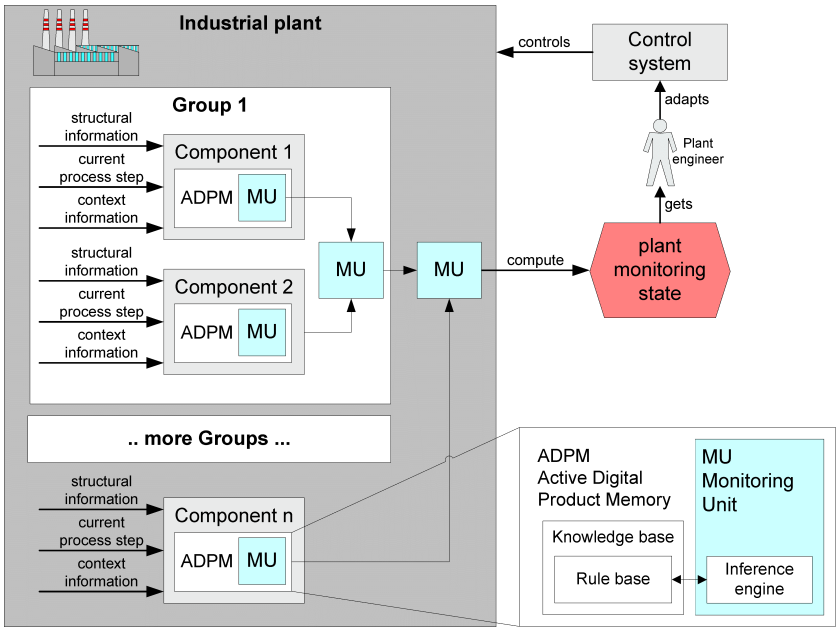


Figure 10.1.: Decentralized Architecture of the monitoring system

plant engineers can optimize the resource efficiency of the entire plant. Thus, the control system of the plant has to be adapted in order to optimize the parametrization of the components and the control procedures.

10.2. Monitoring Model

The knowledge-base described in the previous Section contains a rule base to compute monitoring states based on input data and knowledge incorporated in the plant models. By combining plant models and the monitoring rules in an overall monitoring model stored in the knowledge-base, the rules can be connected with plant knowledge directly. This allows for advanced maintenance and engineering features to be implemented.

The aim of this section is to introduce an umbrella model for the monitoring purpose named ***monitoring model*** which is used to relate all models required for monitoring and for defining basic monitoring concepts. In traditional software engineering, a distinction is made between analysis and design stage of knowledge-based models. These two stages of the monitoring models are presented within the subsequent subsections. First, the analysis monitoring model defined in the analysis stage is presented in Subsection 10.2.1 and the design model is presented in Subsection 10.2.2. Then the usage of structural, process and context knowledge for resource monitoring is discussed in Subsection 10.2.3, 10.2.4, and 10.2.5 respectively.

10.2.1. Analysis Stage

This section introduces fundamental concepts needed to establish an understanding for the monitoring model in a first analysis stage

Monitoring flow

To monitor the resource usage of plant components several sequential steps need to be performed by the monitoring system as shown in Figure 10.2. Monitoring process steps are executed by specific actors. These actors transform the initial sensor data throughout the entire monitoring process into a component state as specified in the following.

The first step of the monitoring flow is to capture the sensor data. In this step, the *sensor* measures a specific physical *quantity* at a certain time to generate a raw *signal* with sensor specific values (e.g. 10 Volt). Then, the signal data is annotated which means that continuous signal data is sampled down to discrete *sensor events*. The quantity value of the signal data is enriched with quantity class, unit and timestamp. In a next step, the state of the component is identified. The *rule engine* uses the numerical value of the property instance to determine the *simple state* of the component monitored by the sensor. In a final step, the same rule engine might infer a *composite state* for a composite component by taking several simple states as input.

Analysis Monitoring Model

The monitoring model as specified in the analysis stage is shown in Figure 10.3. It specifies a *Monitoring Element* which can either be a Component, a Process or a Product

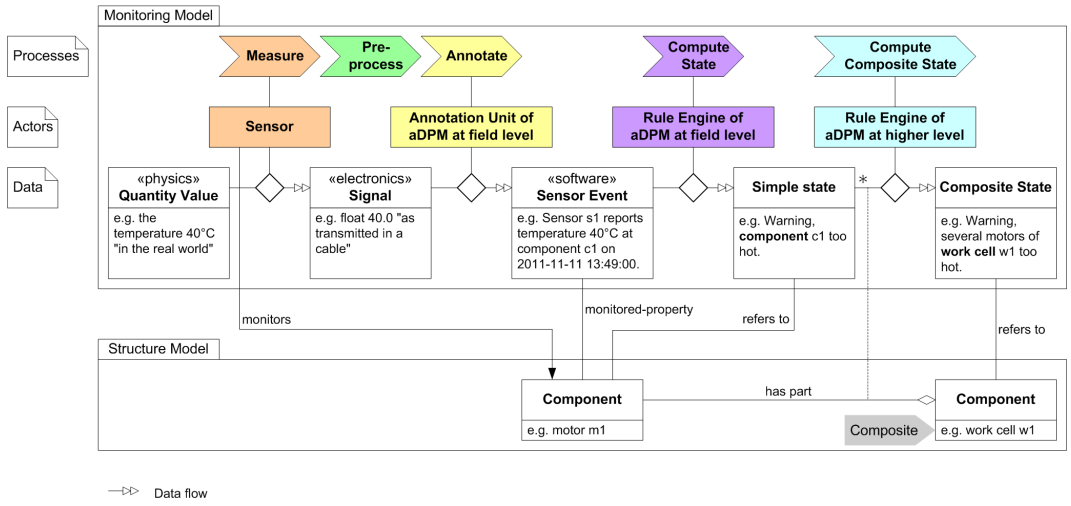


Figure 10.2.: Monitoring Flow

Instance. A *State Vector* refers to this monitoring element and stores Attribute Instances of the Monitoring Element. A *Simple Rule* computes *State Categories* based on the State Vector and different *Contexts*, such as the Activity or the Resource Price. The State Category is composed of *State Instances* of the Monitoring Element with specific *Severities*. State Vector and Category together define the ***State*** of a Monitoring Element. This State can be used by a Composite Rule to compute a State of a composite component such as workcell wc1 which contains several simple components. A detailed description of all concepts in the analysis monitoring model is given in the following.

10.2.2. Design Stage

Based on the fundamental concepts introduced in the analysis stage, the monitoring model in the design stage is derived from the analysis monitoring model. The goal of the design stage is to define a monitoring model which can be implemented in a real plant setting.

The resulting design monitoring model established in the design stage builds on modeling patterns of available model sources, e.g. the Process Specification Language (PSL) as described in Section 8.3, and domain models that arise from our monitoring approach such as the structure model defined in Section 8.2. Industrial monitoring systems should consider at least two types of domain knowledge: plant knowledge (e.g. connections between plant components, products that are produced by the plant), and rules for state recognition (e.g. a rule for inferring the state “powerTooHigh” for motor m1).

A distinction between several *partial plant models* is being made here. Figure 10.4 presents how these partial plant models build upon each other. The resource monitoring framework consists of the structure model, the process model and the state recognition

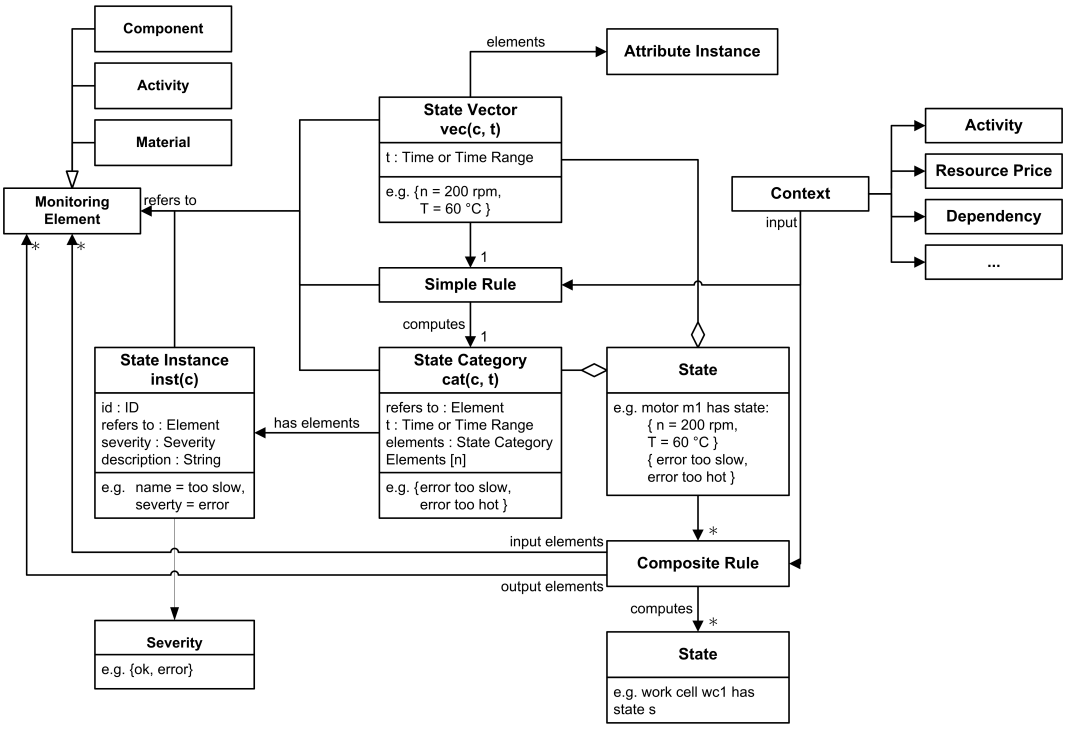


Figure 10.3.: Monitoring Model specified in the analysis stage

model introduced in the subsequent section. The state recognition model depends on the explicit knowledge represented in the structure and process model to simplify the engineering and to verify constraints. Thus, a distinction between the plant model, composed of the process model and structure model, and the state recognition model is made in the following.

State Recognition Model

The state recognition model specifies concepts needed to define rules for the recognition of states for monitored elements in a plant. This model plays a key role in the RMS since it allows for engineering and maintaining monitoring rules in a common manner.

The concepts of the state recognition model are presented in Figure 10.5 and specified in Table 10.1. These concepts support the engineering and maintenance of monitoring rules based on the information stored in the plant models. Similarly to the rule ontology *RuleToOnto* proposed by [104], this model represents information about the rule components and their structures, but the purpose is different. Another similar concept for defining rules in the manufacturing area by means of predefined building blocks was established by [142], but in contrast to this work, they focus on production rules for control systems. The

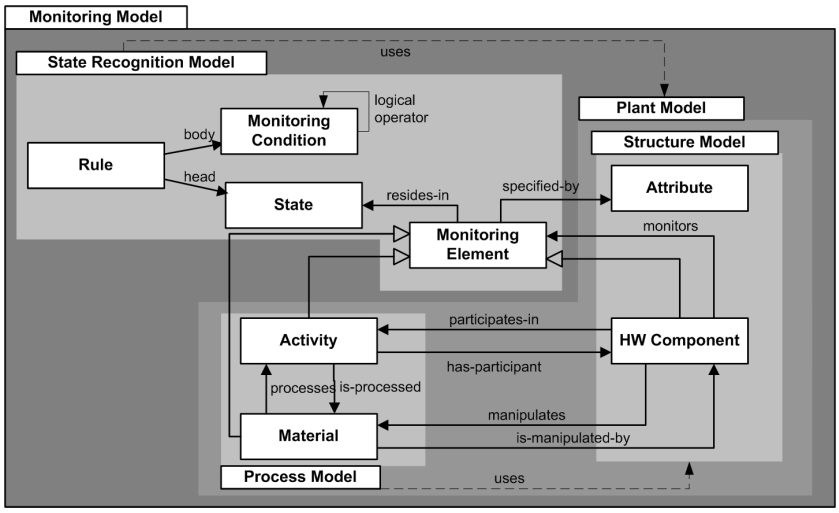


Figure 10.4.: Overview of monitoring model consisting of several partial plant models

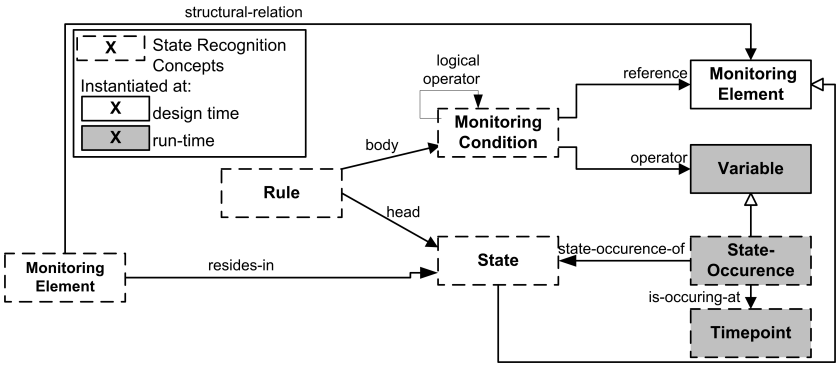


Figure 10.5.: State recognition model as UML diagram

following principles are defined for usage of the concepts of the state recognition model:

- **Principle 1:** A Rule can have multiple bodies, but only one head.
- **Principle 2:** An element can only be declared as a Monitoring Element, if the reference *monitors* is pointing on it.
- **Principle 3:** A State-Occurrence is the *state-occurrence-of* a single ***State***.
- **Principle 4:** If a Monitoring Element *resides-in* a State that is the *head* of a Rule, then this Rule *refers-to* the according Monitoring Element.
- **Principle 5:** If a Monitoring Element *is-specified-by* an Attribute, then the Attribute Condition of the Monitoring Element needs a reference to the same Attribute.

Concept	Informal Definition
Monitoring Element	Central concept of this model that includes all elements of a plant that can be *monitored*. It is either an ***Activity***, a ***Component*** or a ***Material***.
Rule	A rule is used to infer a ***State*** for a ***Monitoring Element*** based on one or more ***Monitoring Conditions***.
Monitoring Condition	A condition which has to be fulfilled to trigger a ***Rule***. It is fulfilled if a specific ***operator*** (e.g. equalTo or biggerThan) holds between an ***Element*** and an ***Occurrence***. Different ***Monitoring Conditions*** can be connected with logical operators such as ***seq*** or ***and***.
State	A class of states a ***Monitoring Element*** *resides-in*. For instance, the state 'Energy Consumption too high'.
State-Occurrence	An occurrence of a state. E.g. 'Energy Consumption of motor m1 too high at 2 PM on May 25, 2013' is a state occurrence of the state 'Energy Consumption too high'.
Occurrence	The super type of all objects that are instantiated at plant operation or run-time.
Timepoint	A point in time.

Table 10.1.: Concepts of the State Recognition Ontology

- **Principle 6:** The Monitoring Element needs to have a *structural-relation* to all Objects that are referenced by the rule that defines the State of the Monitoring Element.

As specified in requirement RQ5 in Section 7.2, the RMS needs to offer mechanisms for a time series analysis, e.g. to identify incorrect process sequences. The state recognition ontology forms a theoretical basis to specify such mechanisms. For this purpose, the relationship *logical operator* was introduced which allows to connect two Monitoring Conditions. Examples for instantiations of logical operator are time series operator such as "seq" (to define a sequence of events) or "during". These operators are commonly used by Complex Event Processing (CEP) rule engines to relate events with each other. Such events play an important role in monitoring systems on process level. A basic definition of event was given by [8]: "An event represents something that occurs, happens or changes the current state of affairs." For RMS an event may signify a threshold exceeding, an information becoming available or a deviation in the process data. Several resource monitoring scenarios can be addressed by means of time series analysis such as:

- reduction of total resource usage for the components based on usage during idle and non-value-added periods
- continuous tracking of maintenance state of components using historical resource usage profiles

- reporting on a per-part basis by accurately accounting for the resource usage of the part being manufactured
- notice emerging trends in the resource usage, such as increased total consumption for successive parts, which may indicate process plant deviations and inconsistencies

Similar to other partial plant models, the state recognition model is specified on Level M2 of the concept model. The concepts *Rule*, *State* and *Monitoring Condition* defined in the state recognition model can be instantiated on M0 or M1 level and used by both, suppliers and plant engineers. An example for the metalevel concepts of the state recognition model and relations to the structure model is shown in Figure 10.6. Most of the rules are implemented on the M1 level, since the rules needed for resource monitoring concern usually multiple components and their attributes. The purpose of resource monitoring is to aggregate resource-relevant information for the entire plant and not for every single components. For example, a rule to verify the energy efficiency of the components requires an additional attribute that specifies the maximum energy efficiency per component.

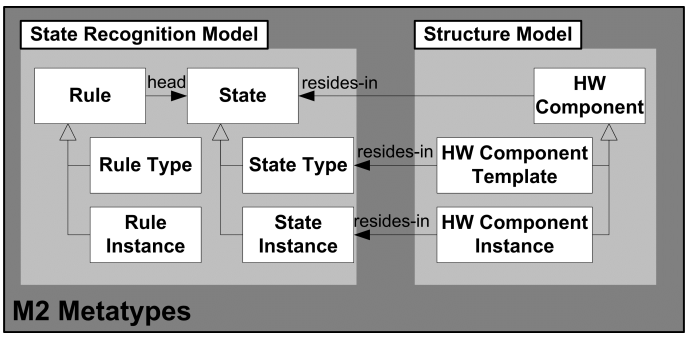


Figure 10.6.: Metalevel concepts of state recognition model

By means of the state recognition model, the appropriate structure of rules can be verified during the engineering phase of the RMS. The advantage over current knowledge-based approaches with plant-specific rules is that the knowledge incorporated in the plant models is used to support the engineering of the rules that are stored in the state recognition model. This allows to improve the engineering process of the rules in different plant engineering phases.

Prior to specifying a rule for a specific monitoring element m1, the engineering expert (plant engineers on level M0 or suppliers on level M1) can for instance query for information about “Siemens Motor1FK7” specified in the plant-specific models, e.g. the attribute “inputPower” or the process “standby” executed by “Siemens Motor1FK7”. During the definition of the rule, the engineering expert can select all elements related to the component “Siemens Motor1FK7” by means of the described modeling approach, e.g. “inputPower” and “standby” of the plant model. This simplifies the construction of a new rule *rule1* based

on the plant knowledge. Then, the system can verify an axiom that says that all monitoring conditions that are bodies of a rule (in this case *rule1*) can only have a *structural-relation* to the objects that the monitored element (in this case Siemens Motor1FK7) refers to.

Additionally, the engineering experts can use the concepts of the state recognition model to manually construct taxonomies of states, monitoring conditions or rules. Such taxonomies offer several benefits: 1) higher reusability of monitoring rules due to common information structures when designing new monitoring rules, reducing the separate investment needed, 2) collaborative working between engineering experts is made easier because automatic selection of objects with identical terms is facilitated as shown in [74].

Subtypes of Concepts of the State Recognition Model

The upper level concepts of the state recognition model can be extended to different applications by adding specific sub concepts, e.g. the condition "temperature of m1 $>$ 40 °C" is represented by an `AttributeCondition` while "activity occurrence of m1 = medium speed" is specified as `ProcessCondition`. In Figure 10.7, the most important domain-specific subtypes of the concepts *Monitoring Condition*, *Element* and *Occurrence* and their relations to concepts of the plant models are shown. For example, *Process Condition* is a subclass of *Monitoring Condition*, *isEqualTo* is a subproperty of *operation* and *Activity-Occurrence* is a subclass of *Occurrence*.

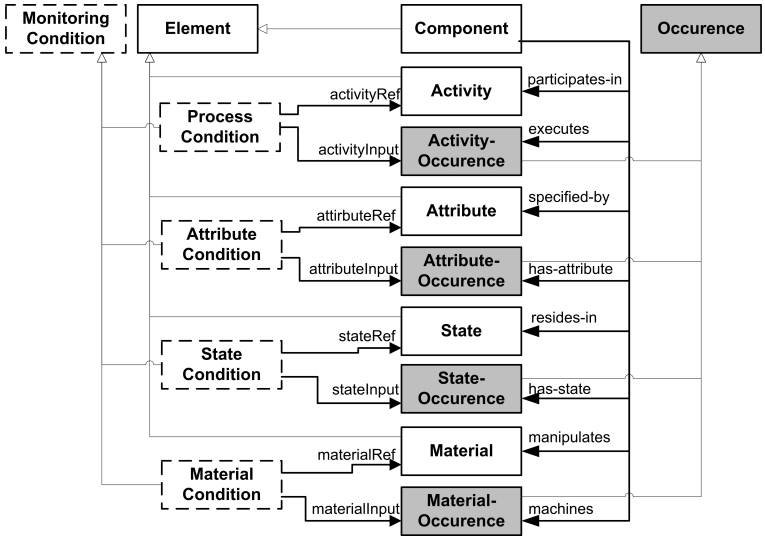


Figure 10.7.: Subtypes of main concepts of the state recognition model

10.2.3. Usage of Structural Knowledge for Resource Monitoring

As found in Section 4, the analysis of sensor data is not sufficient for resource monitoring. The assessment of sensor pressure or temperature change along cables or pipelines to the surrounding components, but also consequences on remote components are required for RMSs. As a consequence, flows such as heat, pressure or energy flows, are needed for resource monitoring. As stated in Section 2.1.1, multiple plant components are currently not equipped with sensors. The modeling of flows allows to assign monitoring states to these components by using the sensor data of surrounding components. An example for usage of structural knowledge for monitoring in this context is the dependence of measurements of surrounding sensors. For instance, two measurements q(c1) and q(c2) of two smart meters *c1* and *c2* in related electrical networks can influence each other. This dependency is expressed by an *energy flow* relationship between the sensors in the structure model. Such a relation allows to compute the influence of an increase in the energy consumption of component *c1* on component *c2*.

In general, monitoring can have two directions of information transfer: The information can either be transmitted in a bottom-up fashion (e.g. an error occurs at a component in the plant and the consequences of the error of the component on the entire plant are computed) or in a top-down fashion (e.g. quality of the paper in a paper mill is poor and the erroneous component has to be identified). To consider both fashions in the RMS, structural knowledge about the containment hierarchy of a plant defined by the relationship "has part" is needed.

Another important part of the structure model needed for monitoring are individual component characteristics and types. The characteristics include default parameters of types, e.g. nominal energy, monitoring thresholds and configuration parameters of components. By using knowledge about the component types, default parameters defined by the manufacturer of the component types can be used by all instances. For example, if 10 instances of a motor of type "Siemens 1FK7" are mounted in a paper mill plant, then the characteristics of this motor specified by Siemens can be reused for all 10 instances.

Usage of Attributes for Resource Monitoring Attributes are also needed for resource monitoring, particularly *monitoring-specific attributes* of components. For the monitoring model, some additional concepts need to be introduced. An overview on all attribute-related monitoring concepts is given in Figure 10.8, while the concepts are outlined in the following in an exemplified manner.

For example, a temperature sensor *ts1* measures the temperature of a motor *m1*. The characteristic *temperature of m1* of the motor is stored in the concept Attribute Occurrence which refers to exactly one HW component. This Attribute Occurrence has a reference to the component motor *m1*, and to the component that measured the value, e.g. the temperature sensor *ts1*.

Attribute Occurrences such as *temperature of m1* are measured with a certain sampling frequency which depends on the sensor settings. At every sampling interval a new *At-*

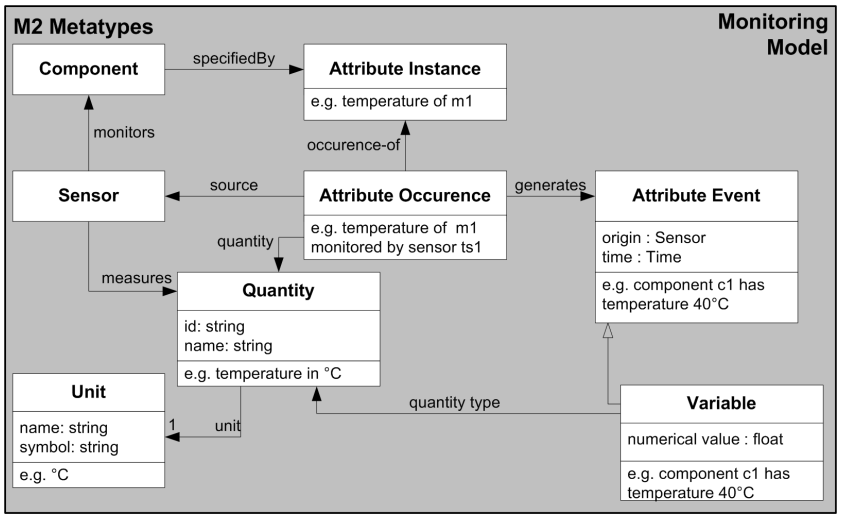


Figure 10.8.: UML diagram of attributes and related concepts

tribute Event is generated. If a sensor measures a signal at a certain time, an ***Attribute Event*** is generated at run-time, e.g. “temperature in °C of *m1*" is an Attribute Occurrence, while “temperature 40 °C of *m1* at 14-01-2012 10:30:00” is an Attribute Event.

Several device-specific Attribute Occurrences have common characteristics. For instance, the attribute *temperature of m1* has the same characteristics as the attribute temperature of a similar motor *m2*, e.g. Unit °C. Since reuse of Attribute Occurrences should be supported, a concept ***Quantity*** is introduced. Quantities are unique, described by an identifier and can be reused for various components.

When the *temperature sensor* measures a numeric value, i.e. 40 °C, this value has to be represented. The concept Variable is thus introduced to represent such sensor measurements. The concept Variable is the most common type of an Attribute Event, e.g. “component m1 has temperature 40 °C”.

The specification of such attributes with related concepts is a fundamental feature of the RMS to assist the plant operators in their decision making task by providing background information about monitoring states. A small monitoring use case is given here to demonstrate this feature. This example scenario consists of two sensors: a temperature sensor and a smart meter (named sentron), used to monitor a motor *m1*. Knowledge provided by the manufacturer and the knowledge engineers can be reused as shown with the following code example of the respective knowledge base:

```
%Knowledge provided by plant engineer at design time
monitors(ts1,m1).
componentType(ts1,temperatureSensor1F7C).
label(ts1,temperatureSensor ts1).
monitors(sentron1,m1).
label(m1,motor m1).
```

```
componentType(sentron1,sentronPAC3200).

%Knowledge provided by manufacturer of sensors
measures(temperatureSensor1F7C,tempC).
measures(sentronPAC3200,powerW).
unit(powerW,watt).
quantity_name(powerW,power).
unit(tempC,degreeCelsius).
quantity_name(tempC,temperature).
```

The rules can use this knowledge to annotate sensor data with additional knowledge. If the sensors used in the examples measure a signal change, e.g. the temperature sensor *ts1* measures 40 °C at time 10:24:30, then a generic rule annotates the sensor data to compute an attribute occurrence of *m1* as follows:

```
RULE annotateSensorData:
   attributeOccurrence(?NumVal,?Unit,?QuantityName,?Sensor,?Comp,?Time)
   ← measures-signal(?NumVal,?Sensor,?Time), monitors(?SensorID,?CompID),
   componentType(?SensorID,?SensorType), measures(?SensorType,?Quantity),
   unit(?Quantity, ?Unit), hasName(?Quantity,?QuantityName),
   label(?SensorID,?Sensor), label(?CompID,?Comp).
```

Applied on the knowledge-base, this rule infers an Attribute Occurrence of motor *m1* with its unit, quantity and sensor. A new fact is then added to the knowledge-base:
`attribute-occurrence(40,degreeCelsius,temperature,ts1,m1,10:24:30).`

Usage of HW Component Templates for Resource Monitoring Another important part of the structure model needed for resource monitoring are HW component templates. HW component templates are defined by the providers of components, usually by the manufacturers themselves. In current monitoring systems the manufacturers role is often neglected. Even though only the manufacturer has the extensive knowledge needed to monitor his components. Therefore, this knowledge is stored within the monitoring model which allows for instance for an automatic threshold monitoring of attributes.

To this end, the manufacturer has to define warning and error threshold for the limits of all type-specific attributes related to the component. These limits are currently stored in the product descriptions of the components, but need to be stored on the M1 level of the plant models to allow an automatic monitoring. An example for such a product catalog is the data sheet of Siemens Simotics S-1FK7 Servomotors as defined in [124]. Once these type-specific attributes are stored in the attributes of the HW component templates. Rules can use these thresholds to monitor the respective components. An example of a synchronous motor 1FK7 manufactured by Siemens with corresponding thresholds is presented in Figure 10.9.

Rules retrieve the threshold stored in the HW component templates, to compute different monitoring states with either severity ok, warning, or error as expressed by the signal lights

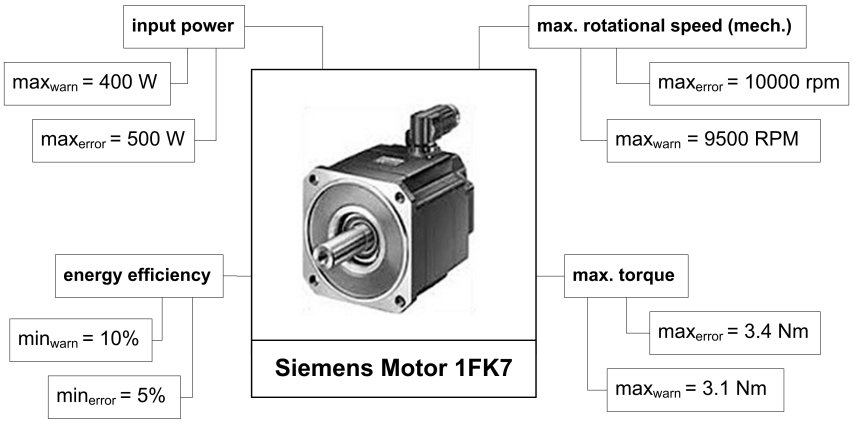


Figure 10.9.: Siemens Motor1FK7 with limits of type-specific attributes

in Figure 10.10. Several generic rules are specified in the monitoring model to automatically compute the correct state:

```
RULE RuleErrorTooLow:
   residesIn(error, TooLow, ?quantity, ?component, ?time) ←
   limitedBy(?attributeInstance, minError), hasValue(minError, ?minVal),
   attributeOccurrence(?numVal,?unit,?quantity,?sensor,?component,?time),
   (?numVal < ?minVal ).

RULE RuleWarnTooLow:
   residesIn(warning, TooLow, ?quantity, ?component, ?time) ←
   limitedBy(?attributeInstance,minError), hasValue (minError, ?minValError),
   limitedBy(?attributeInstance,minWarn), hasValue (minWarn, ?minValWarn),
   attributeOccurrence(?numVal,?unit,?quantity,?sensor,?component,?time),
   (?minValWarn > ?numVal >= ?minValError ).

RULE RuleOK:
   residesIn(oK, , ?quantity, ?component, ?time) ←
   limitedBy(?attributeInstance,minWarn), hasValue (minWarn, ?minVal),
   limited-by(?attributeInstance,maxWarn), hasValue (maxWarn, ?maxVal),
   attributeOccurrence(?numVal,?unit,?quantity,?sensor,?component,?time),
   (?maxVal > ?numVal >= ?minVal ).

RULE RuleWarnTooHigh:
   residesIn(warning, TooHigh, ?quantity, ?component, ?time) ←
   limitedBy(?attributeInstance,maxWarn), hasValue (maxWarn, ?maxValWarn),
   limitedBy(?attributeInstance,maxError), hasValue (maxError, ?maxValError),
   attributeOccurrence(?numVal,?unit,?quantity,?sensor,?component,?time),
   (?maxValError > ?numVal >= ?maxValWarn ).
```

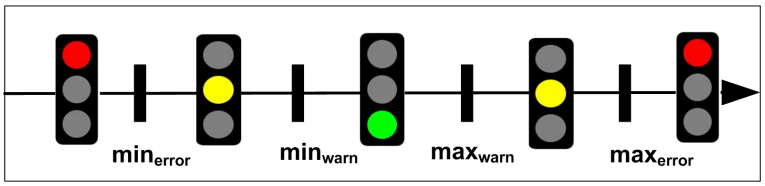


Figure 10.10.: Monitoring states depending on thresholds

```
RULE RuleErrorTooHigh:
   residesIn(error, TooHigh, ?quantity, ?component, ?time) ←
   limitedBy(?attributeInstance,maxError), hasValue (maxError, ?maxVal),
   attributeOccurrence(?numVal,?unit,?quantity,?sensor,?component,?time),
   (?maxVal <= ?numVal).
```

10.2.4. Usage of Process Knowledge for Resource Monitoring

If an erroneous component states occurs, the operator has to decide how to redesign the control system of the plant to avoid such errors in the future. It is not sufficient to warn the operator if an error occurs. The task of the monitoring is furthermore to inform the operator about the process the component is currently executing to improve the redesign of the control system. Thus, process information stored in a process model as described in Section 8.3 is required to compute monitoring results with background knowledge to optimize the processes accordingly.

To determine the monitoring state of components, the current process of the component or remote components is considered by the RMS. The required process information can be divided in two parts: a model of the production process and the dynamic information about the current process state. Today, this information is contained implicitly in the control procedures of the process control devices like programmable logic controllers (PLCs). Since the control procedures of these controllers are generated on a low implementation level, where the individual binary in- and outputs are processed, the code is complex and monolithic [99]. Due to these problems, the control procedure lacks of clarity, comprehensibility and adaptability and should be developed on a higher abstraction level [138]. Therefore, this procedure should be developed on a higher abstraction level as defined by the conceptual modeling approach in Chapter 8.1.

This means that functions of the hardware components are represent as activities, which form the building blocks for the execution of the production process. To get an executable control procedure, the activities are arranged within a process logic in a formal representation. The process logic contains activities and transitions between these activities with links to the respective components. Thus, the model of the production process stored in the knowledge base can be derived directly from this process logic. During run-time, the

control system has to indicate the current active process step and provide process values and dynamic information to the monitoring system.

An important information concerning processes is the duration of an activity. To determine the duration, an ideal production process has to be observed when the plant is commissioned. The duration of every activity of the entire production process has to be measured precisely and then stored in the attributes of the concept activity. This allows for an automatic monitoring of the resource "time" by the RMS. This is illustrated by means of the following two examples.

Example 1 (Process): A robot *r1* can execute multiple activities, e.g. pick, move, place, executed by the control unit. The robot *r1* is driven by a motor *m1*. Depending on the current activity of the robot, the energy consumption of *m1* varies. When the *r1* moves a product, its energy consumption is higher than during picking or placing. Therefore, the monitoring thresholds of *m1* need to be adapted depending on the current activity of the robot. To determine the monitoring state of *m1* and *r1*, the current activity of robot *r1* has to be taken into consideration. This example shows that the monitoring state of a component may depend on an activity executed by a related component. This knowledge is stored in the plant models and thus accessible by the RMS.

The rules needed for this example have the following format:

```
RULE ProcessRule ErrorPowerConsumptionExceedance:
   residesIn(m1, errorPowerTooHigh) ← executes(r1,?ProcessOcc),
   isEqualTo(?ProcessOcc, move), input-power(m1,?AttributeOcc),
   (?AttributeOcc > 10kW).

RULE ProcessRule WarningPowerConsumptionExceedance:
   residesIn(m1, warnPowerTooHigh) ← executes(r1,?ProcessOcc),
   isEqualTo(?ProcessOcc, pick), input-power(m1,?AttributeOcc),
   (?AttributeOcc > 1kW), (?AttributeOcc <= 10kW).

RULE ProcessRule PowerConsumptionOK:
   residesIn(m1, okPower)← input-power(m1,?AttributeOcc),
   (?AttributeOcc < 1kW).
```

Example 2 (Process): Consider a transportation plant, which transports a product on two conveyors *cv1* and *cv2*. Both conveyors are running with the same constant speed to avoid de- or acceleration forces to act on the product. The two conveyors are driven by the engines *m1* and *m2* respectively. These engines include power modules *pm1*/*pm2* to vary their rotational speed, which is controlled by control units *cu1*/*cu2*. The entire transportation unit is grouped in the functional drive group *g* as shown in figure 10.11.

The drive group *G* can execute three different activities: a) drive forward, b) drive backward, c) stop. In this application scenario, the process knowledge is used to monitor the standby voltage V_s of the group *g*. When the motor is in activity "stop" a small standby

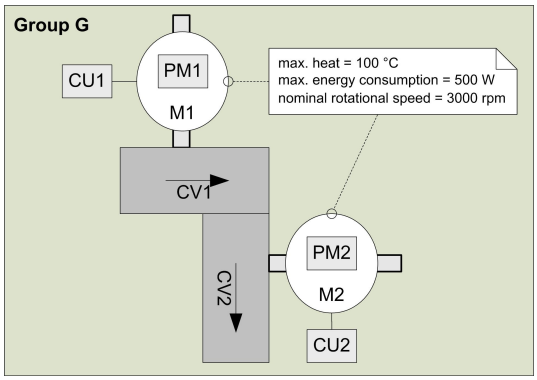


Figure 10.11.: Transportation scenario with conveyors cv1, cv2, which are driven by engines m1, m2 including power modules pm1, pm2 and control units cu1, cu2

voltage is acceptable. If the energy consumption *P* of the two motors *m1* and *m2* is higher than the standby voltage, then the state of the group *g* is set to “error”. The following two rules are defined to compute these states depending on the current activity:

```
RULE ProcessRule ErrorStandbyPowerConsumptionExceedance:
   residesIn(g,errorStandbyPowTooHigh) ← executes(g,stop),
   input-power(m1,?inputPowM1), input-power(m1,?inputPowM2),
   (?inputPowM1 +?inputPowM2 > 16W)

RULE ProcessRule StandbyPowerConsumptionOK:
   residesIn(g,oKStandbyPow) ← executes(g,stop),
   input-power(m1,?inputPowM1), input-power(m1,?inputPowM2),
   (?inputPowM1 +?inputPowM2 <= 16W)
```

10.2.5. Usage of Context Knowledge for Resource Monitoring

To monitor the resource consumption of a production plant or its components, it is not sufficient to observe just the actual consumption values in the context of a process as discussed in Chapter 7. An assessment whether the measured values are within acceptable range often depends on additional knowledge, for example what kind of product the machine is currently producing or what the average energy consumption was for the last products. To determine the current resource situation of the plant, this additional information has to be evaluated continuously and has to be represented in the underlying monitoring rules.

The context definition of [39] is used within this thesis where context is defined as “any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application.”. A monitoring system can make more intelligent decisions and determine

more accurately the situation of the monitored item if additional information (=Context) is provided about all entities that are in some way relevant to the current monitoring task.

Example 3 (Context): A special feature of the transportation plant shown in Figure 15.12 is that the operator of the plant can choose between two production contexts: a) produce with lowest energy consumption, b) produce with minimum delivery time. If the operator chooses the first option (*low energy*), the engines' rotational speed is adapted to reach the maximum energy efficiency and the two conveyors run at a lower speed. If the operator chooses the second option (*min delivery time*), the engines' rotational speed is at its maximum.

Depending on the context ("*low energy*" or "*min delivery time*)" the total energy consumption P_g of the group should not exceed a predefined threshold.

Based on these descriptions, several rules are defined: If the energy consumption P of the two motors *m1* and *m2* differ, then the state of the group g is set to "error":

```
RULE ContextRule errorPowerDifference:
residesIn(g,error) ← input-power(m1,?inputPowM1),
input-power(m2,?inputPowM2), ?inputPowM2 ≠ ?inputPowM1
```

Depending on the context ("*low energy*" or "*min delivery time*)" the total energy consumption P_g of the group should not exceed a predefined threshold of 720W:

```
RULE ContextRule errorInputPowerTooHigh:
residesIn(g,errorPowTooHigh) ← currentContext(g,lowEnergy),
input-power(m1,?inputPowM1),input-power(m2,?inputPowM2),
(?inputPowM1 + ?inputPowM2 >720W)
```

Prioritization of Resources

As shown in the previous subsection, resource monitoring requires context knowledge about industrial plants. An important context is also how the companies prioritize results of resource performance measurements. A company may for instance choose to prioritize other than economic criteria. This view has aroused increasing research interest as stated in [70] and is thus addressed in this subsection.

Consider the following example: The aim of "Ultra low CO2 Steelmaking" (ULCOS) is to reduce the CO2 emission of the furnace of steel plants as depicted in [109]. At a first glance, this seems to be an efficient solution to optimize the resource usage of steel plants. But this technique needs pure oxygen produced by a compressor consuming a huge amount of electrical energy. On the one hand, the emission of greenhouse gas can be reduced with this technique, but on the other hand it offers no clear benefit from resource optimization perspective. The task of a resource monitoring system is to check whether the resource usage of the plant and its components is optimal. In the case of the ULCOS system, several aspects have to be considered to compute the resource efficiency

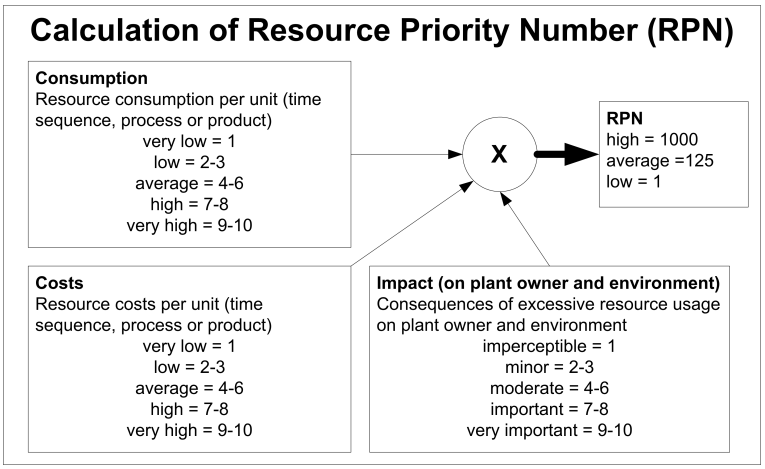


Figure 10.12.: Detailed formula for Resource Priority Number

state of the entire plant. Consequently, the optimization of the resource usage requires an integrated solution that allows to prioritize the resources used by an industrial plant.

Therefore, a systematic technique is defined within this thesis to determine a ***Resource Priority Number (RPN)*** for analyzing the priority of resources. The RPN is defined in a similar manner than the Risk Priority Number in *Failure Mode and Effects Analysis (FMEA)*. FMEA is an analysis method used to become familiar with an unknown plant [63]. It was primarily developed to identify and prevent potential failures in a system before reaching the customer. Usually, an FMEA procedure starts in the plant design phase and consists of three main parts: failure analysis, risk analysis, and counter-measures. In the risk analysis part of an FMEA, the analyst, or normally a team of analysts, rates each failure with a number between one and ten in terms of the following aspects: the probability of occurrence, the severity concerning the consequences, and the probability of detection. The product of these three assigned values results in a Risk Priority Number for each component. The Risk Priority Number is not a probability; it simply prioritizes all possible failures similar to the Resource Priority Number, which prioritizes the resources used by an industrial plant. The RPN can then be used by a control system to schedule processes based on the priority of the resources consumed by the processes or by monitoring systems that compare different resources with each other.

The RPN is also a number between one and ten, but it is determined based on: consumption, cost, and impact. The product of these three assigned values results in a Resource Priority Number used to prioritize the resource usage of every plant component for producing a product or executing a process. The detailed formula is depicted in Figure 10.12. This number has to be determined for every resource used by an industrial plant based on experiences and measurements.

To determine the consumption of a resource, a unit has to be chosen by the experts. For

instance, the plant expert chooses a process step executed by a component as basic unit to compute the RPN. To define the *consumption*, the different resources consumed by a component need to be compared based on this basic unit, e.g. an hydraulic drive has a very high consumption of the resource “oil” during processes executed by the drive while an electrical drive has a very high consumption of the resource “electrical energy”. The *costs* are calculated based on the current prize for a resource. Once calculated they are valid for all components, e.g. the current energy price is 0.5€per kWh. The third input value is the resource usage *impact*, which is determined either based on the KPIs of the company (e.g. the resource “time” for time-sensitive orders has a higher priority than the resource “energy”) or industrial standards that compare the shortage of natural resources can be taken into consideration as well. Depending on the resource, the input values are calculated in a specific manner. Examples are shown in Table 10.2.

Resource	Consumption	Cost	Impact
Energy	unit per Product or per Process	(price per unit) x (energy efficiency)	KPIs or resource index as defined by standards
Component Waste	deviation from optimal operating point p_{opt}	costs for deviation from optimal operating point per component	KPIs or resource index as defined by standards
Time	deviation from reference time and time for optimal process t_{opt}	costs for deviation time per process	KPIs

Table 10.2.: Examples for calculation of Resource Priority Number

10.3. Rule Maintenance

Difficulties arise in rule-based systems, when contradicting or redundant rules are generated and when the number of rules exceeds a certain limit that the experts cannot cope with. Therefore, the following section demonstrates how the previously defined monitoring model can be used for maintaining the monitoring rules contained in the monitoring model.

An exemplary monitoring workflow established for motor *m1* demonstrates this difficulties in Figure 10.13. This workflow shows how multiple sensors (e.g. temperature sensor, smart meter) measure resource-related attributes of motor *m1*, while the plant is in an energy-efficient context and the *m1* is executing the process "run". This input information is assessed and processed by multiple rules to compute a certain amount of resource-related monitoring states. This example shows that for one single motor, a huge amount of rules and states is specified which have to be maintained and structured.

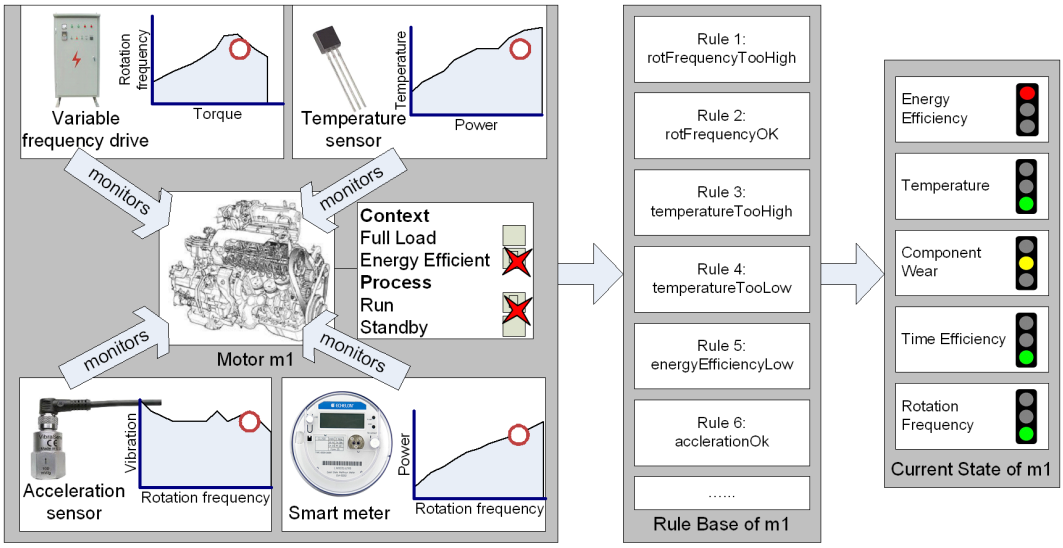


Figure 10.13.: Monitoring Workflow of component Motor m1

Different maintenance tasks, such as verification, classification and querying were realized in the context of this thesis by means of Semantic Web Technologies, such as OWL DL reasoning and SPARQL queries. For implementing the maintenance examples, Protege was used as ontology framework, HermiT and Pellet as OWL DL reasoner and SPARQL as query language. An example is provided for every maintenance task and the realization of this example is described in detail in the subsequent subsections. Further, several additional purposes which can be supported within the same task are mentioned.

A representative example for an instantiation of the monitoring model used throughout this section is shown in Figure 10.14. The shown rule says that "If a Motor *m1* executes

an activity that is equal to *runFast* and the measured value *InputPower* is bigger than 400 and m1 is part of the component workcell 1 with ID *wc1* which currently resides in the state *tempTooHigh*, then the state of the motor *m1* is set to *powTooHigh_RunFast_TempTooHigh* which means that the power consumption and the surrounding temperature are too high during the process runFast". In widely used rule notations with variables denoted with '?', we can express this exemplary monitoring rule as presented in the gray box.

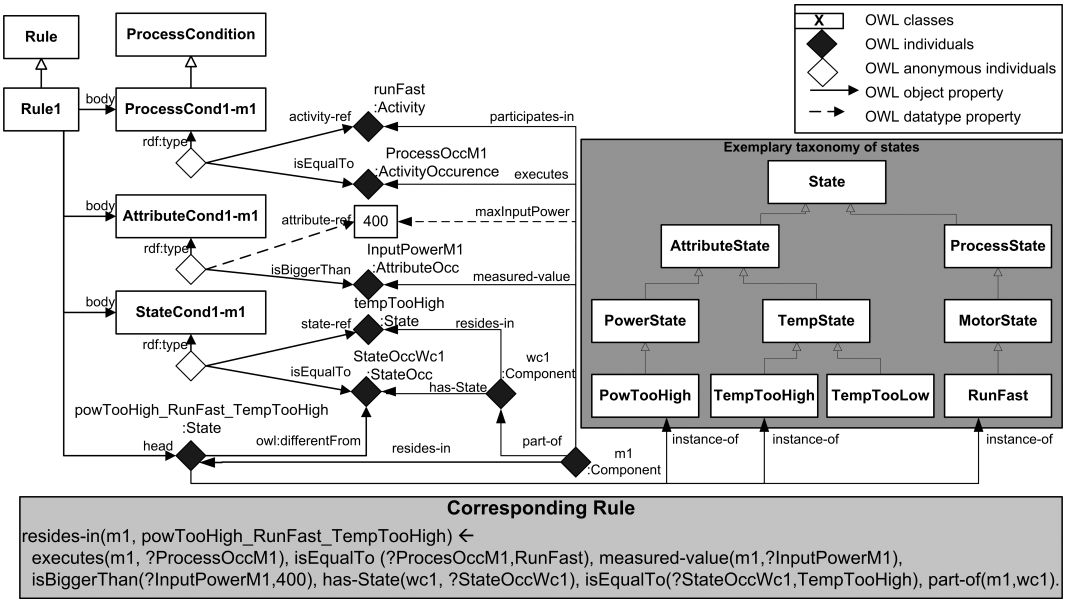


Figure 10.14.: Example for usage of state recognition model

10.3.1. Verification of Rules

Various engineering experts are usually involved in the implementation of a monitoring system for an industrial plant. They define rules to compute the individual monitoring states of the plant elements that have to be monitored. During the design of the monitoring rules or during later adaptations redundant, contradicting or inconsistent rules might occur. The authors of [89] note that it is thus important to check *validity* of rules written in an abstract syntax of the rule languages. Our knowledge-based approach allows the engineering experts to verify the consistency of their monitoring rules by identifying the erroneous rules with OWL DL reasoning mechanisms.

The purpose of the following verification example is to identify contradicting rules. Two rules are contradicting if they infer distinct Monitoring States and take the same Monitoring Conditions as input. For example, `Rule1` infers the state `tempTooHigh` and `Rule2` infers the state `tempTooLow`, while both refer to the same Monitoring Condition

m1attributeCondition. Whereas states tempTooLow and tempTooHigh are defined as different objects. If the two models are combined in one model and an OWL-DL reasoner, e.g. Pellet [128], is executed, then the models become unsatisfiable. Consider two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ expressed in OWL DL syntax shown in the following for this example:

$$
\begin{aligned}
\mathcal{O}_1 \;=\; & \{\mathit{Rule} \sqsubseteq = 1\ \mathit{head} \\
& \mathit{Rule}1 \sqsubseteq \mathit{Rule}\ , \\
& \mathit{Rule}1 \equiv \exists\ \mathit{body}.\ \mathit{m1attributeCondition}1\ , \\
& \mathit{Rule}1 \sqsubseteq \exists\ \mathit{head}.\ (\{\mathit{tempTooLow}\})\ \} \\
\mathcal{O}_2 \;=\; & \{\mathit{Rule2} \sqsubseteq \mathit{Rule}\ , \\
& \mathit{Rule2} \equiv \exists\ \mathit{body}.\ \mathit{m1attributeCondition}1\ , \\
& \mathit{Rule2} \sqsubseteq \exists\ \mathit{head}.\ (\{\mathit{tempTooHigh}\}) \\
& \{\mathit{tempTooHigh}\} \neq \{\mathit{tempTooLow}\}\}
\end{aligned}
$$

Another possible verification task is to identify rules that can never be executed. This is the case if rules use inverse Monitoring Conditions. For instance, if the Monitoring Condition attributeCondition1 is defined as "Temperature of M1 $>$ 20" and the Monitoring Condition attributeCondition2 is defined as "Temperature of M1 $<$= 20". If these two conditions are both bodies of the same rule, then a monitoring state cannot be computed.

10.3.2. Classification of rules

During implementation of a rule-based monitoring system, a large amount of monitoring rules is generated. The users can usually not cope with such a huge amount of rules without some sort of structuring. Therefore, the classification mechanism offered by OWL reasoners are used to structure the rules and the Monitoring Conditions in various classification hierarchies. Further, some general classification purposes with according examples are defined.

One purpose of the classification is to structure the rules by identifying subtypes of Monitoring Conditions. Generally speaking, a rule *Rule1* is classified as a subtype of another rule *Rule2*, if it uses a subset of Monitoring Conditions of the rule *Rule2*. An example for a realization of this classification task is if the monitoring rule Rule1 as presented in Figure 10.14 has three Monitoring Conditions, while another rule Rule2 has only two of the three conditions. Rule1 is thus classified as a subtype of Rule2 by an OWL Reasoner. The according OWL ontologies $\mathcal{O}_3$ and $\mathcal{O}_4$ are defined as follows in OWL DL syntax:

$$
\begin{aligned}
\mathcal{O}_3 \;=\; & \{\mathit{Rule}1 \sqsubseteq \mathit{Rule} \\
& \mathit{Rule}1 \equiv \exists\ \mathit{body}.\ \mathit{m1StateCondition}1 \\
& \qquad \sqcap \exists\ \mathit{body}.\ \mathit{m1attributeCondition}1 \\
& \qquad \sqcap \exists\ \mathit{body}.\ \mathit{m1processCondition}1 \\
& \mathit{Rule}1 \sqsubseteq \exists\ \mathit{head}.\ (\{\mathit{powTooHigh\_RunFast\_TempHigh}1\})\} \\
\mathcal{O}_4 \;=\; & \{\mathit{Rule2} \sqsubseteq \mathit{Rule} \\
& \mathit{Rule2} \equiv \exists\ \mathit{body}\ \mathit{m1attributeCondition}1 \\
& \qquad \sqcap \exists\ \mathit{body}.\ \mathit{m1processCondition} \\
& \mathit{Rule2} \sqsubseteq \exists\ \mathit{head}\ (\{\mathit{powTooHigh}\})\}
\end{aligned}
$$

Another purpose of the classification task is to identify redundant rules. Therefore, we check whether some rules in the monitoring rule base have equivalent bodies (Monitoring

Conditions) and infer the same States. These rules are then specified as equivalent using the OWL construct `owl:equivalentClasses`.

10.3.3. Querying of rules

For the construction of new monitoring systems or for the adaptations of existing monitoring systems, engineering experts want to reuse previously defined monitoring rules. To allow reuse of rules, the experts have to be supported in querying the plant-specific knowledge base. We take advantage of the previously specified monitoring model to realize these queries. In our user scenario, the plant engineer wants to exchange all motors of type M1FK7 against new motors of type M1PH8 with a lower energy consumption. He needs to identify all rules that can be reused by identifying all rules that are defined for motor of type M1FK7 and refer to the state “PowerState”. A taxonomy of states defined by the plant experts can be used for this query since it specifies all instances of the state “PowerTooHigh”. SPARQL queries under OWL entailment regime [53] expressed in Turtle syntax are used for this task:

```
SELECT ?rule ?component
WHERE {
     ?component rdf:type pl:M1FK7.
     ?rule rdfs:subClassOf mon:Rule,
           [ rdf:type owl:Restriction ;
             owl:onProperty mon:head ;
             owl:someValuesFrom [owl:oneOf (?state)]
           ].
      ?state rdf:type ?stateType.
      FILTER (?stateType = PowerState)}
```

Queries can further be used to filter meta-level information, e.g. to identify all rules that have a high severity, or for post-monitoring analysis with data-analysis tools.

11. Evaluation of scalability

In recent years, semantic applications based on ontologies have become increasingly important. But scalability still remains one of the major drawbacks in using ontologies for practical applications. The scalability is dependent on the performance of OWL DL reasoners used for the rule maintenance task proposed in the previous section. But which reasoner has the highest performance and can be used to accomplish this task? This question can be assessed by finding an answer to research question RQ3:

> **Research Question 3:** What is the influence of an increase in different scalability factors (e.g. plant size or sampling rate) on the scalability and performance of a resource monitoring system realized by means of knowledge-based technologies?

One aspect of RQ3 is the scalability of the RMS. This aspect can be tested by using the monitoring model for the rule maintenance task. To evaluate this aspect of the RQ, the evaluation question E3a was answered in this Section:

> **Evaluation Question 3a:** Is the monitoring model proposed for the rule maintenance task scalable for reasonably large industrial plants?

Performance tests for evaluating the scalability of the monitoring model for increasing numbers of monitoring rules were thus performed. As primary performance measure, the response time for reasoning was used, i.e. the time that is needed for a given maintenance task. Another interesting measure for performance could be the utilization of system resources which was not considered here. Empirical tests should thus primarily serve as a proof of concept, showing that the approach scales, with acceptable computational demands, to reasonably large industrial plants. Experiments were carried out on a 2.4 GHz Intel Core PC with 2,92 GB memory with Protege 4.3. The common OWL DL reasoners HermiT, Fact++ and Pellet were used for the reasoning task.

A number of performance evaluations for OWL reasoners have already been performed in the past. Based on a broad evaluation reported by [16], the scalability limits of the maintenance task were first estimated. The outcome of this report was that common OWL DL reasoners can process ontologies up to 30.000 axioms. This means that theoretically up to 1200 rules stored in the monitoring ontology can be processed by these reasoners since approximately 25 axioms have to be added to the ontology for every rule and the

initial ontology consists of 100 axioms. But since the amount of classes is not stable for the monitoring model as in the examples provided in [16], additional evaluation tests were performed to determine the limits of scalability in practice.

The initial monitoring model consists of 23 classes and a total amount of 124 axioms, while the number of classes and axioms of the monitoring model increases with the amount of monitoring rules and components introduced. For every rule, an average amount of 21 axioms is added to the ontology. In this test framework, four test ontologies were defined: 1) the reference ontology `REF` described in Section 10.3.2 which contains only one component and one rule, 2) the ontology `CAS` of a representative plant described in [27] composed of 9 components and 20 monitoring rules, 3) the ontology `RESCOM` of a conveyor plant build for demonstration purposes within the RES-COM project composed of 43 components and 108 rules, 4) the ontology `SRO1` corresponds to a model of a reasonably large industrial plant and consists of 320 rules and 130 components.

Ontology	**Axioms**	**HermiT** time in ms	**Fact++** time in ms	**Pellet** time in ms
REF	137	47	15	47
CAS	550	562	94	554
RESCOM	2194	10.719	7.458	689.661
SRO1	6.032	1276.302	1806.748	10110.085

Table 11.1.: Performance Results of scalability tests

The results of the performance tests are shown in Table 11.1. The proposed maintenance approach scales for ontologies up to 5000 axioms which corresponds to approximately 300 rules with an appropriate reasoner such as HermiT or Fact++. These results showed also that HermiT performs better than the other reasoners. The classification time of HermiT was 20 minutes for the ontology `SRO1` which proves that the approach scales for reasonably large industrial plants. The classification task is the most time consuming task, but it has to be performed only once at the end of the engineering phase. The high computation time for large industrial plants is thus acceptable.

Part IV.

Implementation Aspects

12. Implementation of a Knowledge-Based Resource Monitoring System

As manufacturing trends impose increasing demands for generic and changeable knowledge-based systems that integrate various kinds of plant knowledge by providing excessive user support, an implementation of knowledge-based RMSs has to respect these demands. The use of knowledge-based models and in particular Semantic Web technologies as underlying technology stack is said to fulfill these implementation demands [161]. Various knowledge-based tools and techniques are currently used in the manufacturing domain, nevertheless an appropriate technology stack for implementing an RMS has not yet been found.

The purpose of this chapter is to specify an RMS under specific implementation aspects. The most relevant aspect is that the system's implementation needs to fulfill the RMS requirements specified in Chapter 7. To this end, a proposed software architecture is specified in Section 12.1. This software architecture is mainly based on a plant model repository and a state recognition module instantiated in Section 12.2 and 12.3 respectively. In an evaluative step described in Chapter 13, the approach is illustrated in more details by discussing the pros and cons of the implementation defined within this thesis in comparison to common knowledge-based tools and procedures. To achieve this, the knowledge-based tools and procedures are evaluated based on the RMS requirements. Furthermore, the usability of the entire system is evaluated in a small user study presented in Chapter 14.

12.1. Software Architecture of the Resource Monitoring System

As stated in Section 10.1, a decentralized software architecture is required for the RMS. Figure 12.1 displays this idea and sketches architectural aspects of the suggested system.

The main constituent of the system is formed by the MU, a software component with the functionality of analyzing plant-relevant input data and of deriving states of plant components from this data and detecting faulty states. An MU is deployed on a *monitoring node*, which can be any suitable run-time environment in the automation field, such as an ADPM running on an embedded controller or an industrial PC. By this design, the functionality of an MU can be placed basically anywhere in the plant.

Inside an MU, a *state recognition module (SRM)* performs the task of recognizing the current monitoring state of an observed plant component by analyzing the provided input, such as sensor signals, process parameters, or state information for other plant compo-

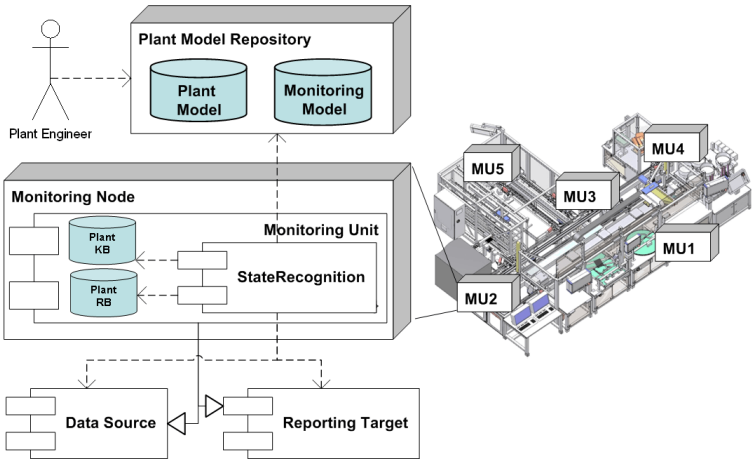


Figure 12.1.: RMS architecture as UML component diagram (⇢ = access, ▷ = generalization).

nents. In a knowledge-based setting, state recognition relies on a knowledge base consisting of two parts. On the one hand, a *plant knowledge base* (plant KB) provides information about static plant model aspects, such as the electrical connections of the plant components. On the other hand, a *plant rule base* (plant RB) hosts the knowledge about how to derive appropriate state information from input data and process parameters of a plant, e.g. in form of declarative rules. This twofold knowledge serves the easy adaptation of state recognition functionality to modifications in the structure of a plant without requiring the re-programming of code.

Any MU has access to *data source* and *reporting target* software components from where it receives its input data and to where it reports state recognition results, respectively. The MU itself inherits the functionality of both these software components, which allows for cascading monitoring and diagnostic units in a hierarchy with one unit reporting its state results as input to another. This design allows for decentralized solutions combining field level data analysis on embedded controllers with the central accumulation of single device and component states for observing the overall plant.

In an initialization phase, an MU connects to the central *plant model repository (PMR)* that hosts various partial plant models maintained by plant engineers. From this repository, it retrieves the content for its plant KB and RB knowledge components as a configuration for the monitoring task it performs. If an MU is set up to observe a conveyor, for example, it retrieves the required parameters of the conveyor component from the plant's structural and process models and associated state recognition rules from the monitoring model. This means that an MU's behavior is determined by information that a plant engineer explicates in the PMR, which provides for a high flexibility of MUs adaptable to declarative plant model information. The PMR and SRM are the most important parts of the entire monitoring system. Suggestions on how to implement this architecture on a semantic technology stack are given in the subsequent sections.

12.2. Implementation of the Plant Model Repository

In the proposed implementation, the PMR is based on a four-layered conceptual modeling approach that adheres to the Meta Object Facility (MOF) paradigm as described in detail in Chapter 8. Various types of plant engineers in different user roles can use this concept model to formulate plant knowledge at their appropriate level of abstraction. Such a Model-Driven Architecture for the representation of layers allows for engineers to participate in the modeling process in a collaborative manner and to ensure a formal representation of the expert knowledge as shown by [75].

Semantic MediaWiki (SMW) [71] with its underlying semantic data storage facilities is proposed within this thesis for instantiating the conceptual modeling approach. SMW provides for collaborative editing of and navigation in structured plant model web pages connected by hyperlinks. Their underlying RDF-based data model allows plant engineers to formulate rich queries that are evaluated in support of inference mechanisms, while plant models can also be exported as OWL/RDF ontologies. In addition, it provides the means to handle all modeling entities of the various models in an information system in terms of storage, visualization and editing.

In Section 12.2.1, use cases derived from experiences made in two research projects are presented to illustrate the issues that occur when defining a knowledge-based PMR for the monitoring of resources in industrial plants. These use cases can be tackled by means of the conceptual modeling approach and SMW as underlying technology stack as shown in Section 12.2.2.

12.2.1. Use Cases for the Plant Model Repository

UC1. Standardization of plant models. A team defines standardized concepts for plant models, e.g. within a company, within a group of researchers or an industrial consortium (IEC, ISO, etc.). They describe the modeling concepts in detail and define mappings between concepts to create a common understanding of the problem domain. Various models of different standards are mapped on each other, e.g. the concepts “Equipment’ and “Material” as defined in VDI 5600 are equivalent to the concept “Resource” as specified in IEC62264.

UC2. Translation between terminologies of engineering experts. Engineering experts use their discipline-specific terms for concepts in the problem domain, e.g. the term “Device” used by mechanical engineers is equivalent to the term “Hardware Component” used by electrical engineers. Because the plant engineers have to communicate with each other and understand each others terminologies, translations are required to automate references to terms and facilitate their communication.

UC3. Provision of semantic device description for plant engineers. The plant engineers want to identify devices with specific attributes, e.g. identify all asynchronous motors with a maximum energy consumption of 500W and a maximum weight of 20kg. Thus, the device specifications of the manufacturer in their product catalogs have to be enhanced

with semantic descriptions including meta information to allow the plant engineers to query on the device data.

UC4. Provision of plant information for industrial automation systems at run-time. The plant operator at run-time needs to understand the meaning of multiple messages from industrial automation systems, e.g. warnings, to determine his next action. If the run-time messages were related to the design-time knowledge about the industrial plant as defined by the plant operation experts, the operators can be supported or some decision can even be automated.

UC5. Check user-defined constraints. The plant owners can define constraints on their industrial plants that have to be respected by the plant operators, e.g. a "digital in" port has to be connected with a "digital out" port. These constraints are validated at design time to guarantee the correct operation of the plant.

UC6. Reuse of plant knowledge. If a plant operator has to exchange a component in a model or reuse parts of a plant model, e.g. working conditions, average energy usage, etc., he wants to reuse the information that was gathered about the plant and its components during its entire life cycle. The advantage of reusing models is that one needs less manual effort to configure a manufacturing system and the implementation can save time to a great extent.

12.2.2. Semantic Mediawiki

SMW forms the basic infrastructure of the PMR. While a conventional wiki includes structured text and untyped hyperlinks, only a semantic wiki is based on the representation of metadata elements which allows for logical reasoning on wiki content. SMW is probably the most popular and mature semantic wiki [71]. It relies on the same wiki engine as Wikipedia and uses constructs from the RDF and OWL to support semantic web features such as reasoning and querying.

The authors of [26] highlight the potential of using such semantic wiki-based tools for the modeling of complex domains composed of various processes and components. An advantage of a model repository implemented within a semantic Wiki is that it allows for the collaboration of several domain experts that are familiar with diverse plant aspects, e.g. the manufacturers of the components can define general monitoring rules for their components in the Wiki. Another benefit of a semantic Wiki is that it provides on the one hand an overall view on all aspects of the monitored plant with its navigation links and on the other hand specific aspects of the plant can be filtered with ASK queries. For instance, a query to identify all plant elements that are not yet monitored. In the following section, special features of the SMW infrastructure needed for the RMS are depicted.

Guided Web-based Editing

First of all, modeling concepts of all levels of the conceptual model (Entities, Relationships and Attributes) are represented by wiki pages. This enables users to create and edit

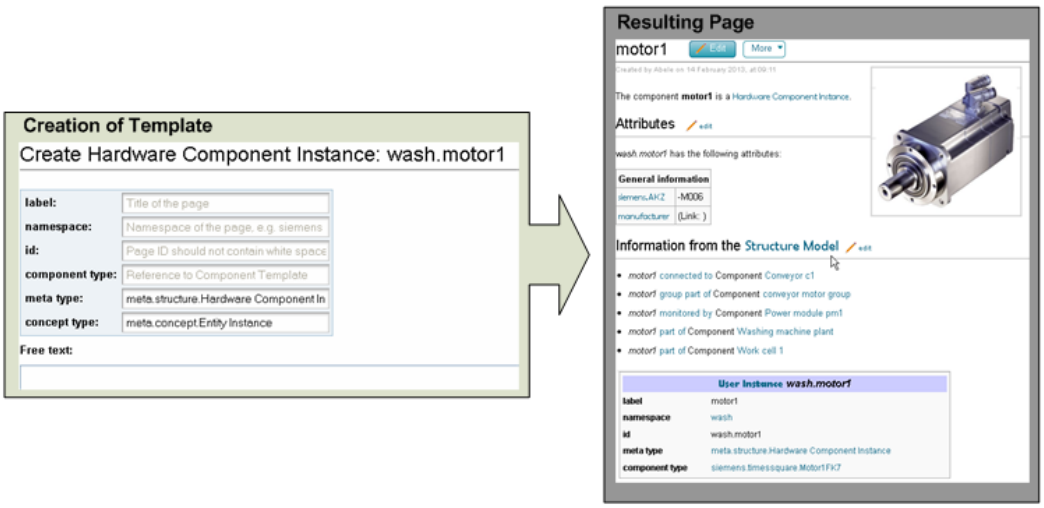


Figure 12.2.: Example for generation of new page “motor1” based on a semantic form

structured plant knowledge in the form of web pages which are stored in underlying RDF models. SMW provides several ways for custom formatting (e.g. HTML, CSS) to be applied to pages. User guidance is provided by templates and an additional extension “Semantic Forms” that allows engineering experts to create and edit plant engineering data using predefined modeling patterns. Semantic forms are pages consisting of markup code which gets parsed when a user goes to add or edit data. For the RMS, the semantic forms are used to instantiate concepts of M3 and M2 level in order to guide the editing of pages used by editors of type or instance models. An example for a semantic form for HW Component Instances is shown in Figure 12.2. It is defined by the knowledge engineers of M2 level and used by plant engineers of M0 level. Several mandatory fields can be defined for all pages using this form. At the same time, forms can use templates that allow to store queries or rules on the pages using the forms.

Additionally, the web-based architecture of SMW supports the collaborative editing of community knowledge (as described in UC1) since web pages are accessible by all members of the community. Editing can be restricted for certain user groups, e.g. authors of a meta model or plant constructors, which allows for a clear separation of the modeling levels.

Semantic Data Representation

SMW allows to use constructs of RDF and OWL. OWL classes are defined by means of category pages with the tag “Category:”, individuals of these classes are normal pages. Pages in SMW can be related via “property” pages, which allows

for entering semantic data in SMW. Entities of the concept model are modeled as usual pages, but Relationships and Attributes are realized with SMW properties. Several special properties can be used to annotate pages, e.g. “Property:Is inverse of” and “Property:Is Equal To” can be used for mapping equivalent discipline-specific concepts on each other as defined in UC2. To relate entities on the four modeling levels with each other, predefined property pages are introduced such as the page with ID “Property:meta.concept.entity type”.

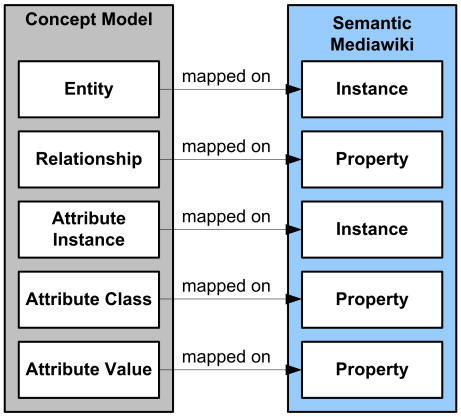


Figure 12.3.: Mapping of concepts

When implementing the PMR in SMW, a decision was made to store all entities and attributes as individuals instead of using categories, as shown in 12.3. This decision was made because the levels M2 and M1 contain concepts that are instances and classes at the same time. Due to this concept reification, there is no clear distinction between classes and individuals. All concepts of the concept model are stored on the SMW instance level. Relationships are defined as SMW properties and allow to distinguish between the four modeling levels.

SMW offers import and export mechanisms for ontologies. Engineering experts using the wiki can export selected plant engineering data via the page “Special:ExportRDF” by entering a list of articles into the input field. The export will contain an OWL/RDF specification with various description blocks for exported elements, which can then be used by industrial automation systems as defined in UC4. An engineering expert can import an existing industrial OWL ontology into the wiki meta-model using a script that translates OWL into wiki syntax.

Adaptable Navigation and Visualization

The SMW infrastructure offers means for interlinking plant engineering model entities. It is thus possible to navigate in the interlinked structure of the models and to visualize the entities on web pages, which can be adapted by the users. The engineering experts can distinguish the different levels of the conceptual modeling approach with namespaces of the pages in SMW. Every page URL starts with a specific namespace. The name of the page is separated with “:” from its namespace. All concepts of the conceptual model are stored in the namespace “meta.concept” and all metamodels have their own namespace starting by “meta”. The namespaces of the type and instance levels are chosen by the users, e.g. the product catalog of Siemens has the namespace “siemens”. The pages with namespace “meta” can be reused for new plants or components as required in UC6. All concepts of the concept model are stored in the namespace “meta.concept” and all metamodels have their own namespace starting by “meta”, e.g. the structure model has

the namespace “meta.structure”, the context model has the namespace “meta.context”, etc. Beyond, namespaces for definitions of terms were introduced, e.g. “en/de” for terms in English or German standards, e.g. “standard.iec61360” and documentation of the wiki content stored in the namespace “doc”.

All objects in SMW can be identified by their specific page Id. These Ids correspond to the page URL and are unique identifiers. Ids have no empty spaces and can thus be used in queries or rules. The property “Property:meta.concept.id” is used to specify the Id of a page.

12.2.3. Reasoning-supported Validation and Querying of Data

Another feature of the SMW infrastructure is that it provides a formal verification mechanism for querying and validating constraints on plant engineering data. A number of wiki modules handle the communication with external tools and knowledge technology components, e.g. a Triple Store Connector (TSC) that allows to use a triple store with advanced querying and reasoning functionality. With such a linked triple store, the engineering experts can define queries in SPARQL syntax, e.g. to identify components with specific attributes as described in UC3. Furthermore, the user-defined constraints on plant engineering data in the wiki can be validated by reasoning over the inserted semantic engineering data, which is needed for the realization of UC5. Another example for validation features that support the plant engineering process is depicted in the subsequent section.

Validation Support within the Plant Engineering Process

The process of engineering an industrial plant as described in Section 8.2 consists of three major steps presented in Figure 12.4. First, it is important to define the role classes to be used in the respective domain. User-specific role classes are defined as specializations of standard role classes from respective standard libraries. A plant hierarchy representing a requirement-based containment view of the intended plant is created with HW component instances assigned to the concrete roles. As a second step, the defined *role class*es are used for selecting suitable components from vendor-specific product catalogs. Therein, available components are defined and detailed by the vendor by means of *HW component templat*es. After manually selecting a suitable *HW component template*, it is assigned to the previously defined *HW component instance*. Due to the manual *HW component template* selection process and the variety of attributes to be considered, *HW component templat*es not fully meeting the role requirements might be assigned. Consequently, a mechanism for identifying such inconsistencies is beneficial. In the third step, the *HW component instance*s are connected by *relationship*s, representing the plant-specific inter-component connections. Thus, the previously defined *plant hierarchy* is completed. Due to the complexity of a plant model, incorrectly connected *HW component instance*s might occur. To identify such inconsistencies as early as possible, *relationship*s need to be checked for properly connected *interface instance*s.

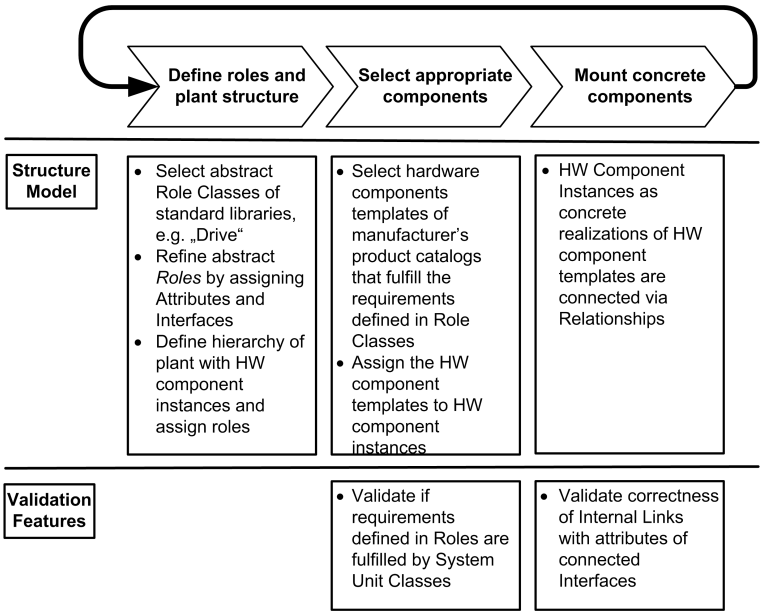


Figure 12.4.: Plant engineering process

In the following, two consistency checks along the plant engineering process implemented with Semantic Web technologies are presented. These examples are extracts of validation features defined in [5]. The firs example shows how to validate role requirements of HW component templates and then, the correctness of interface connections is verified.

Validate Role Requirements of Internal Elements During the plant engineering process, appropriate HW component templates for the HW component instances based on the requirements defined in the according roles are manually selected. An issue of this selection task is that the operator is often confronted with an enormous list of requirements. It might thus happen that requirements of a component instance as defined by its role class are not entirely fulfilled by its template. To support the operator, role class requirements need to be validated by checking the attributes of all role classes and HW component templates assigned to a HW component instance. Thus, a SPARQL query was defined to identify all HW component instances where the attributes do not match. The SPARQL queries under OWL entailment regime expressed in OWL functional-style syntax have the following format:

```
SELECT (?a ?role ?valR ?ci ?valCI) WHERE {
ClassAssertion(ObjectSomeValuesFrom(:supportsRole ?role) ?ci)
SubClassOf(?role ObjectSomeValuesFrom( :requiresAttribute ObjectIntersectionOf(
 DataHasValue(:hasValue ?valR) DataHasValue(:hasName ?a))))
SubClassOf(?suc ObjectSomeValuesFrom( :hasAttribute ObjectIntersectionOf(
 DataHasValue(:hasValue ?valCI) DataHasValue(:hasName ?a))))
```

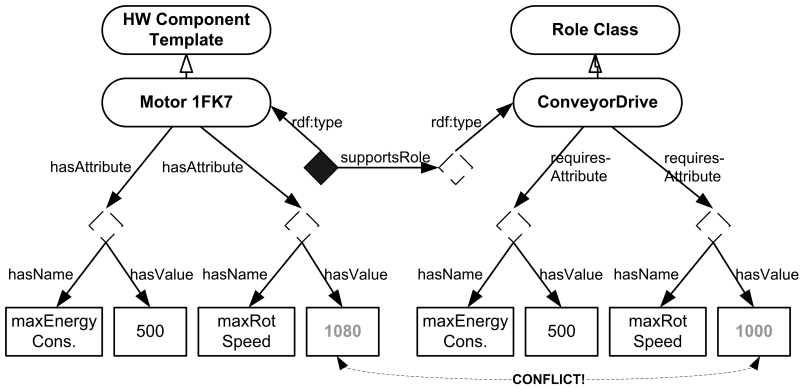


Figure 12.5.: Example for validation of attribute consistency

```
FILTER( ?valR != ?valCI)}
```

In Figure 12.5, an example is demonstrated. The attributes of HW component template *m1FK7* are validated, which is an instance of *Motor1FK7* and supports the role requirements of the *role class ConveyorDrive*. This *role class* requires an attribute *maxEnergyCons* with a value *500 W* and an attribute *maxRotSpeed* with a value of *1000 rpm*. *Motor1FK7* supports only the required value of attribute *maxEnergyCons*, a conflicting attribute is *maxRotSpeed* since its value *1080* is too high for the *ConveyorDrive* (Note: a higher speed of the conveyor belt may cause serious damage).

A similar query can be defined to validate the attributes of *interface instance*s required by *role class*es.

Validate Correctness of Internal Links User-defined component libraries play an important role for the development of exchangeable plant models in formats such as AutomationML. These libraries provide an extended vocabulary with additional semantics. Such semantics offers a possibility of being validated. A system for an automatic verification of such links has also been established in [1].

An example for such a library for interfaces is defined by AML and named *AMLBaseInterfaceLib*. This library defines SignalInterfaces (SIs) with an attribute “Direction”, and possible values “In”, “Out” or “InOut”. According to the AML standard, SIs with the direction “In”/“Out” can only be connected to SIs with opposite direction or “InOut”. SIs with the direction “InOut” can be connected to SIs of arbitrary direction. In order to validate correct wiring, this semantics needs to be defined formally.

To support operators that use the SI attributes of the *AMLBaseInterfaceLib*, all SIs with direction “In” or “Out”, which are connected to an *interface class* of the same direction

[1] European Patent EP1958101B1 (Fay, A. and Drath, R.): “System and Method for the Automatic Verification of Planning Results”, 2006

have to be identified. The ontology $\mathcal{O}_3$ is used for this purpose in SMW with the following structure:

$\mathcal{O}_3$ = SubClassOf(*SignalInterfaceIn* ObjectIntersectionOf(*SignalInterface*
ObjectSomeValuesFrom(*:hasIntAttr*
ObjectIntersectionOf(DataHasValue(*:hasValue* "In") DataHasValue(*b:hasName* "Type"))))) ,
SubClassOf(*SignalInterfaceOut* ObjectIntersectionOf(*SignalInterface* ObjectSomeValuesFrom(*:hasIntAttr*
ObjectIntersectionOf(DataHasValue(*:hasValue* "Out") DataHasValue(*b:hasName* "Type")))) ,
EquivalentClass(*MiswiredSignalInterface* ObjectUnionOf(ObjectIntersectionOf(ObjectSomeValuesFrom(
:connectedTo SignalInterfaceIn) *SignalInterfaceOut* ObjectIntersectionOf(
ObjectSomeValuesFrom(*:connectedTo SignalInterfaceOut*) *SignalInterfaceIn*))))

Based on $\mathcal{O}_3$, individuals, that are inferred to be instances of the *MiswiredSignalInterface* class, can be identified as incorrectly wired. The *interface instance*s are classified based on their direction attribute to the *SignalInterfaceIn* or *SignalInterfaceOut*, respectively. Since the *interface instance* ii1, which has the direction “In”, is linked to *interface instance* ii2, which has the same direction, ii1 is additionally identified as *MiswiredSignalInterface*.

12.3. Implementation of the State Recognition Module

In this section, instructions how to implement the SRM based on the state recognition model specified in Section 10.2.2 for engineering and deploying the monitoring rules are given. SMW was used for the engineering of monitoring rules, Protege as an ontology editor, and HermiT, Pellet, and Fact++ as OWL DL reasoners. The consistent integration of the tools in the engineering process is presented in the following.

Due to the complexity of industrial plants, the amount of monitoring rules accumulated over time can be huge. This is the reason why we focus on how to engineer, maintain and deploy the accumulated rules. The implementation architecture of the SRM is presented in Figure 12.6 and consists of a **Rule Engineering**, a **Rule Maintenance** and a **Rule Deployment** part.

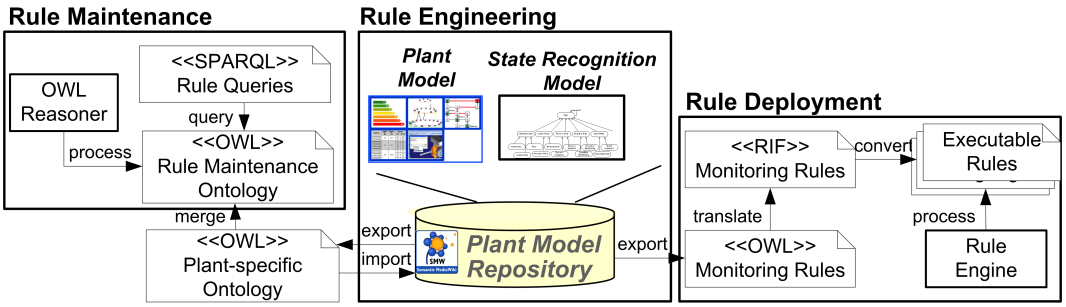


Figure 12.6.: Architecture of the SRM

The plant engineers use the Rule Engineering part to store knowledge about their plants in the PMR as shown in Section 12.2. Selected parts of the plant-specific model required for resource monitoring can be exported via the page “Special:ExportRDF”. The resulting ***Plant-specific Ontology*** will contain an OWL/RDF specification of the exported elements. The Rule Maintenance part builds on this Plant-specific Ontology by merging it into the ***Rule Maintenance Ontology***. The Rule Maintenance Ontology is required to complement the RDF file with axioms and classes needed for the rule maintenance tasks as described in detail in Section 10.3. An OWL Reasoner processes the resulting ontology and executes specified maintenance tasks, e.g. verification of rules by identifying semantically incorrect rules or classification of rules to arrange them in a structured taxonomy. The plant engineers can further define ***SPARQL Rule Queries*** for the monitoring model, e.g. to filter reusable monitoring rules. In the Rule Deployment part, the plant-specific state recognition model defined on level M0 of the PMR is transformed into a concrete rule language format in a systematic way. A similar problem was also solved by [17], but they transformed an OWL-DL ontology directly into a rule base of Jess. In contrast to this approach, we allow the users to choose between different rule languages and thus convert the ontology into a generic Rule Interchange Format. First, all rule-related knowledge of

the PMR is exported and stored in ***Monitoring Rules*** in OWL according to the monitoring model specified in Section 10.2. Then, the OWL Monitoring Rules are transformed into Monitoring Rules represented in RIF XML serialization syntax. This syntax can be translated into various other rule language formats. The plant engineer benefits from the conversion in RIF format since he can select an appropriate rule engine depending on plant-specific constraints. For example, if the monitoring system should support real-time events, he can use a complex event processing rule engine such as Drools Fusion [106].

12.3.1. Rule Engineering

In the Rule Engineering part, the plant engineers store knowledge about their plants in the PMR as shown in Section 12.2. The reusability is guaranteed by the PMR since it contains the upper ontologies that support the engineering experts in the task of defining plant-specific monitoring models based on the existing ones. The upper ontologies are represented in OWL/RDF axioms in SMW. The process model based on PSL is for instance transformed in OWL as specified in [35]. Plant engineering experts have the possibility to either add their additional concepts to the model repository or match their concepts to existing repository concepts, which guarantees reconfigurability. SMW supports domain experts in their task of modeling plant engineering data and monitoring rules according to the concept model and allows them to visualize and navigate plant engineering model entities.

Another feature of SMW are templates and semantic forms, which allow for specifying explicit monitoring rule patterns for the engineering experts. The software automatically determines inconsistencies and redundancies in the rule base, similar to an approach proposed by [110], where the rules are declared by means of Semantic Query-Enhanced Web Rule Language (SQWRL). An example for such a rule pattern in SMW is demonstrated in Figure 12.7. Using this rule pattern, the experts need to choose a plant and a component first. An important advantage of the Semantic Forms extension is that it supports auto-completion, which enables the software system to show a drop-down list of possible completions when the user starts typing. These completions can be restricted on semantic properties or pages of specific type, e.g. the "'Monitored Attribute"' field in our example shows only attributes related via the property "'specified-by"' to the component "'workcell1"' selected in the first step.

The exemplary monitoring rule of the plant-specific model is then exported and merged into the monitoring knowledge-base. In natural language this rule says: "'*workcell1* resides in the state *powerTooHighRunFast* if the measured power consumption of *workcell1* is bigger than *500 Watt* and the current *activity* of is *runFast*"'. This exemplary monitoring rule can be expressed as:

```
RULE powerTooHighInProcessRunFast:
   residesIn(workcell1, powerTooHighRunFast)← (workcell1Power > 500),
   (workcell1ActivityOcc = runFast).
```

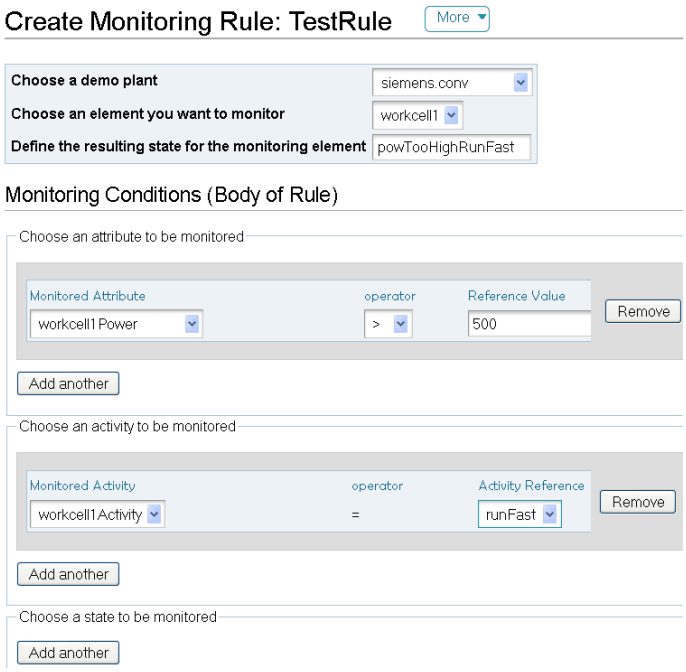


Figure 12.7.: Example for rule modeling pattern in SMW

The according rule uses predefined plant knowledge of the structure model, e.g. the attribute *workcell1Power*, and the process model, e.g. the activity *runFast*, while SMW verifies if the provided knowledge is consistent, e.g. by checking if the activity *runFast* is assigned to the component *workcell1*. The condition "workcell1Power $>$ 500" is represented by an `AttributeCondition` while "workcell1ActivityOcc = runFast" is specified as `ProcessCondition`. The same exemplary rule expressed as OWL DL axiom is represented as:

$$
\begin{aligned}
\mathcal{O}_1 = \{ & \mathit{Rule1} \sqsubseteq \mathit{Rule} \\
& \mathit{Rule1} \equiv \exists\, \mathit{body}.\, \mathit{AttributeCondition1} \sqcap \\
& \qquad \exists\, \mathit{body}.\, \mathit{ProcessCondition1} \\
& \mathit{Rule1} \sqsubseteq \exists\, \mathit{head}.\, (\{\mathit{powerTooHighRunFast}\}) \\
& \mathit{AttributeCondition1} \sqsubseteq \mathit{AttributeCondition} \\
& \mathit{AttributeCondition1} \equiv \mathit{hasValue.attributeRef}\ '500' \sqcap \\
& \qquad \exists\, \mathit{biggerThan}.\, (\{\mathit{workcell1Power}\}) \\
& \mathit{ProcessCondition1} \sqsubseteq \mathit{ProcessCondition} \\
& \mathit{ProcessCondition1} \equiv \exists\, \mathit{activityRef}.\, (\{\mathit{runFast}\}) \sqcap \\
& \qquad \exists\, \mathit{equalTo}.\, (\{\mathit{workcell1ProcessOcc}\})\}
\end{aligned}
$$

Another important aspect of our decentralized system architecture is an establishment of cascading MUs that report their state results as input to other MUs. This is realized by means of a `StateCondition` to allow comparison of monitoring states and a `StructuralCondition` to define conditions on structural relations between components. An example for a rule using these two conditions is `Rule2`: "A component is set to *pow-*

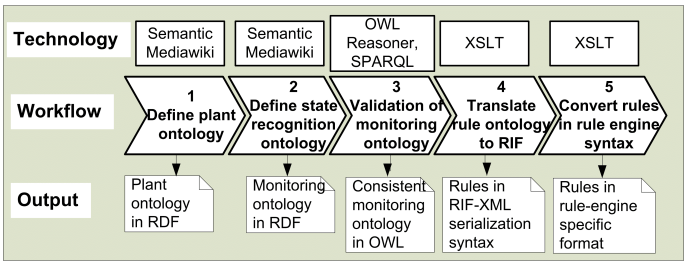


Figure 12.8.: Deployment Workflow

TooHighError if it is partOf *workcell1* and if *workcell1* resides in the state *powTooHighRunFast*". This rule is represented by $\mathcal{O}_2$.

$$
\begin{aligned}
\mathcal{O}_2 = \{ & \mathit{Rule2} \sqsubseteq \mathit{Rule} \\
& \mathit{Rule2} \equiv \exists\, \mathit{body}.\, \mathit{StructuralCondition1} \sqcap \\
& \qquad \exists\, \mathit{body}.\, \mathit{StateCondition1} \\
& \mathit{Rule2} \sqsubseteq \exists\, \mathit{head}.\, (\{\mathit{powerTooHighError}\}) \\
& \mathit{StructuralCondition1} \sqsubseteq \mathit{StructuralCondition} \\
& \mathit{StructuralCondition1} \equiv \exists\, \mathit{equalTo}.\, (\{\mathit{workcell1}\}) \sqcap \\
& \qquad \exists\, \mathit{structuralRef}.\, (\{\mathit{hasPart}\}) \\
& \mathit{StateCondition1} \sqsubseteq \mathit{StateCondition} \\
& \mathit{StateCondition1} \equiv \exists\, \mathit{state-ref}.\, (\{\mathit{powerTooHighRunFast}\}) \sqcap \\
& \qquad \exists\, \mathit{equalTo}.\, (\{\mathit{workcell1State}\})\}
\end{aligned}
$$

12.3.2. Rule Deployment

Several steps are necessary to implement the final monitoring system in a real plant. Figure 12.8 shows a workflow indicating how to get from a general plant and monitoring ontology stored in a Semantic Mediawiki to executable rules that can be processed by a rule engine at plant operation. The first three steps were described in detail in previous Sections. Step 4-6 are not covered in a similar degree of detail and are briefly summarized in this chapter.

The result of step 3 is a consistent monitoring ontology containing the plant-specific monitoring rules. To make these rules executable, they have to be mapped on the syntax of a rule language. The engineering experts should not be limited to one predefined rule engine, therefore the rules defined in the monitoring ontology are translated into the generic Rule Interchange Format (RIF) [2].

In step 4, each monitoring rule is translated by XSLT with an appropriate XSL style sheet to W3C RIF XML serialization syntax, according to the mapping as defined in the W3C RIF standard document [3]. All rules are processed by an XSLT processor, e.g. Saxon

[2] *RIF Basic Logic Dialect*, W3C Rule Interchange Format (RIF) Working Group, 2005, available at http://www.w3.org/TR/2013/REC-rif-bld-20130205/.

[3] *RIF in RDF*, W3C Rule Interchange Format (RIF) Working Group, 2005, available at http://www.w3.org/TR/rif-in-rdf/.

[4], parametrized with an XSL style sheet, containing the XSLT translation specific to the RIF XML syntax. The XSL style sheet consists of a number of XML templates, identifying matching XML structures and specifying appropriate RIF-specific code fragments these XML structures will be translated to.

In step 5, each monitoring rule is translated via XSLT from RIF XML serialization syntax to a specific rule engine language format. During XSLT processing, matching XML structures are retrieved and translated to rule code fragments. An example for an executable rule in the Etalis rule language that corresponds to the example in Section 12.3.1 is provided in the following:

```
resides-in(m1, powTooHigh_RunFast_TempTooHigh)<-
    resides-in(wc1,StateOccWc1) and
          executes(m1,ProcessOccM1) and
          measured-value(m1, InputPowerM1)
    where (ProcessOccM1 = m1RunFast,
                    StateWc1 = tempTooHigh,
              InputPowerM1 > 400, has-part(wc1,m1)).
```

When the deployment workflow is completed, a monitoring run-time module can execute the rules. The rule engines have to be configured manually to indicate how the variables contained in the rule ontology are instantiated at run-time and how to access the plant knowledge as data base of the rule engine, e.g. by built-in functions. The references to interfaces defined in the plant ontology can be used for the configuration, e.g. a reference to a sensor interface or a request to the control unit to get the current process step. Then the rule engine generates monitoring events and annotates them with a machine-readable semantics. These events are either presented to the user or processed by further modules (e.g. a diagnosis module).

[4] http://saxon.sourceforge.net/

13. Evaluation of RMS Requirements

In Chapter 7, a comprehensive list of requirements facing RMS in modern manufacturing environment was proposed. The implemented RMS has to meet these requirements entirely. Thus, research question RQ4 needs to be answered:

> **Research Question 4:** What tools and procedures can be used to meet the RMS requirements and to improve ease-of-use through knowledge-based techniques to support plant engineering experts during the engineering process of a Resource Monitoring System?

Different tools and procedures for implementing the plant model repository and the state recognition module were compared with the implementation proposed in the previous chapter to answer this question. It has to be verified if these tools and procedures fulfill the RMS requirements specified in Section 7 by discussing their pros and cons. The RMS requirements were evaluated in test scenarios within the RES-COM project and a literature review. To evaluate RQ4, the evaluation question EQ4a was answered in this Section:

> **Evaluation Question 4a:** Does the implemented plant model repository and state recognition module meet the RMS requirements?

First, the PMR is evaluated in Section 13.1, then the SRM is discussed in a similar manner in Section 13.2. Finally, a summary of the evaluation is given in Section 13.3.

13.1. Plant Model Repository

Wide-spread modeling tools and standards of Semantic Web, industrial and software engineering research communities were selected for a comparison with the implementation described in the previous section. All evaluated approaches are open source and consist of an editor, e.g. AutomationML, for user interaction and an underlying metamodel, e.g. CAEX, to structure the plant knowledge. For this evaluation, the implementation of the PMR described in the previous section is referred as Approach *M.a* while other implementation alternatives are denoted by *M.x* (M = modeling). The results of the evaluation are shown in Table 13.1.

Approach M.a. As described in Section 12.2, the approach defined within this thesis allows to define partial plant models on the metalevel and use general plant concepts such as “Monitoring State”. As found in Chapter 9, the system is universally applicable and satisfies RQ1 entirely. Research on industrial control systems has shown that a model-driven approach as proposed here addresses dimensions such as modularity, flexibility, extensibility, reusability and interoperability [140], which facilitates reconfigurability as set out by RQ2. By instantiating concepts defined on higher levels of abstraction, engineering experts can adapt or extend the plant knowledge-base according to their needs. Hereby, RQ2 is fulfilled wholly. RQ3 is satisfied to a high degree by the conceptual modeling approach, since the concept model allows to define and relate the partial plant models in a unified way from the conceptual model. This allows for integrating various kinds of knowledge about the plant (e.g. process, structural and context knowledge) and for an integrative view on an industrial plant across the respective engineering disciplines. Furthermore, the concept of namespaces for Wiki pages offer a clear separation of different plant aspects. All partial plant models have their own namespaces and the namespaces of the type and instance levels can be customized.

To meet RQ6, SMW templates and Semantic Forms extension are used to allow engineering experts to specify explicit modeling patterns for the metalevel and libraries of devices or processes. An automated verification of the models on all levels with regard to their conformity with the concept model and predefined modeling patterns is constantly executed by SMW. To achieve this, a number of SMW modules handle the communication with external tools, e.g. a Triple Store Connector (TSC). With such a linked TSC, users can define their own queries in SPARQL or rules with the SMW rule extension, which allows for satisfying RQ7 completely by detecting design errors in all stages of system design. SMW allows also to define links between plant model entities at different levels of abstraction and to navigate within a plant model by means of hyperlinks, e.g. to navigate from a component instance "‘Motor m1"’ to its type “Siemens1FK7”. Complete fulfillment of RQ8 is accomplished by the ability of SMW to define SPARQL queries, which can be promoted on specific pages by predefined navigation style sheets and allow user-defined navigation between modeling entities.

Approach M.b. The first alternative for implementing the PMR was to use Protege [97], the most widely used ontology development tool, instead of SMW. Our conceptual model is then used as underlying metamodel for capturing plant and rule knowledge with the Protege editor. Due to the usage of the conceptual model, the core requirements are completely met as explained in the previous paragraph. Since Protege was mainly developed for experts of the semantic web community, extensive user guidance is not provided. For instance, RQ6 is fulfilled only to the extent that Protege allows to define reusable libraries and forms, but it does not offer user templates for defining modeling patterns in a uniform way. RQ7 is satisfied in part by additional reasoner plug-ins, e.g. Pellet, that perform consistency checks on plant knowledge, however additional rules or checks need to be formulated by the user. Besides, there is not yet an appropriate visualization extension for navigating within huge knowledge bases [129] as demanded by RQ8.

Approach M.c. The standard data exchange format CAEX was also compared with our approach. CAEX is application-independent, and thus meets RQ1. It follows an object-oriented paradigm and allows to integrate various kinds of plant knowledge in alignment to its metamodel, however it is based on a fixed metamodel, which cannot be adapted. RQ2 can thus not be satisfied. The plant knowledge has to be stored physically in a unique XML file, which hinders effective collaboration of engineering experts. Due to missing concept reification, there are no formal mechanisms for consistent aligning of modeling constructs concerning different plant aspects and RQ3 cannot be fulfilled entirely. A supportive user interaction tool for handling CAEX files is the AutomationML editor [41]. This editor satisfies RQ6 only partially by offering reusable libraries of devices or interfaces, but it lacks templates or forms for supporting the engineering experts in defining reusable modeling patterns on abstract metalevels as basis for consistency checks as requested by RQ7. Another drawback of the editor is that RQ8 cannot be fully met, since automatic navigation by means of hyperlinks is not provided, although links between entities are displayed.

Approach M.d. Another model-based engineering approach that was considered as an alternative is proposed by [45]. Their approach extends CAEX with an additional domain model developed for the concrete manufacturing domain and a validation mechanism using MathML. It is similar to our approach since it follows a model-driven engineering paradigm, but in contrast to our approach, the domain model is instantiated on XML-based technologies, which hinder extension and enhancement of models in general as found in [95]. RQ2 can thus not be met entirely. Nevertheless, they provide a formal description for integrating different plant aspects, which allows for satisfying RQ3 partially. Besides, RQ6 and RQ7 are fulfilled to the extent that they define modeling patterns and consistency checks for modeling data, but the modeling patterns are predefined and cannot be modified or enhanced by the engineering experts. In summary, user guidance is improved compared to Approach *M.c*, but still the navigation ability as defined in RQ8 cannot be satisfied entirely.

Approach M.e. Another alternative is SysML, a recent language derived from UML and dedicated to systems engineering applications [141]. An advantage of SysML is the high-level metamodel concepts, which ensure a broad applicability of this approach and allow to fulfill RQ1. But these metamodels rely on an object oriented architecture (three level architecture), which impedes definition and reconfiguration of partial plant models on a higher level of abstraction as demanded by RQ2. The SysML metamodel ensures that RQ3 can be satisfied with respect to the integration of different modeling constructs, however the mapping between knowledge of different plant aspects has to be accomplished for every plant separately. RQ6 is partly fulfilled by the reusable libraries that can be defined with SysML, however templates or forms for supporting the engineering experts in their modeling task are not available. The consistency checks of SysML are based on high-level modeling constructs and cannot be extended by the user, which prevents the complete fulfillment of RQ7. According to [55], RQ8 cannot be met since SysML does not provide a holistic understanding of the system's structure and behavior due to insufficient navigation and visualization ability.

	RQ	*M.a* SMW + Conceptual Model	*M.b* Protege editor + Conceptual Model	*M.c* AutomationML editor + CAEX	*M.d* AutomationML editor + CAEX & MathML [45]	*M.e* SysML editor + SysML
CORE RQS	RQ1 (application independence)	allows to define metamodels for various domains	being RDF based there is a generic metamodel	can be applied on all industries	metamodel is fine-tuned for discrete manufacturing industry	system models can be applied to any application
	RQ2 (reconfigurability)	additional concepts can be added to the metamodel	additional aspects can be added to the metamodel	XML-based technologies hinder extension and enhancement of models as found in [95]	XML-based technologies hinder extension and enhancement of models as found in [95]	high level metamodel, no incorporation of adaptable partial plant models
	RQ3 (integration)	allows different experts to integrate various kinds of knowledge	conceptual model ensures that partial plant models are engineered in an integrated manner	no metadata that describes how partial plant models relate to each other	formal description how to integrate different plant aspects is defined	physical mappings between partial plant models can be established
OUTER CORE RQs USER GUIDANCE	RQ6 (modeling patterns & libraries)	libraries, templates and forms support modeling of plant knowledge	libraries can be defined, but no templates for modeling patterns	libraries can be defined, but no templates or forms for patterns	libraries can be defined, but no templates or forms for patterns	libraries can be established, but no templates or forms for patterns
	RQ7 (consistency checks)	rules for consistency checks can be added, automatic checks available	reasoning on plant knowledge to perform checks by using plug-ins	XML schema validation, but no semantic or user-defined checks	predefined and user-defined consistency checks are available	no modeling patterns as basis for consistency check
	RQ8 (navigation)	good navigation ability supported by navigation style sheets	for large plants, the navigation ability is inadequate [129]	although being displayed the links cannot be accessed automatically	although being displayed the links cannot be accessed automatically	for large plants, the navigation ability is inadequate [55]

Table 13.1.: Evaluation of approaches for the PMR, (gray cells = RQ not fulfilled, blue cells = RQ partially fulfilled, white cells = RQ entirely fulfilled)

13.2. State Recognition Module

Various wide-spread rule formats and languages of semantic web, industrial and software engineering research communities were considered for a comparison with the implementation of the SRM proposed in the previous Section. All these rule formats and languages are open source and consist of an editor, e.g. Prolog, for user interaction and an underlying meta language, e.g. Etalis, with its rule engine to define the structure of the monitoring rules. For this evaluation, the implementation of the SRM described in the previous section is referred as Approach *R.a* while other implementation alternatives are denoted by *R.x* (R = Rule). A summary of the evaluation is represented in Table 13.2.

Approach R.a. As presented in Section 10.2, a monitoring ontology is used in combination with SMW and Drools as rule engine to address the core requirements. An application-independent state recognition ontology is used for modeling rules. Hereby, RQ1 is fulfilled. This rule metamodel is stored in SMW and can be adapted by the plant engineering experts which entirely satisfies RQ2. To meet RQ3 entirely, seamless integration is possible by linking from the state recognition model into the partial plant models.

To address the state recognition functionality requirements, Drools was chosen as rule engine. RQ4 and RQ5 are fully satisfied since Drools uses ”if-then"-like rule constructs as a basic knowledge representation paradigm, next to complex algorithmic calculations as well as special time-related operators, such as e.g. `seq` or `during`, for analyzing the time correlation of monitoring events. To fulfill RQ6 completely, user guidance is provided for the experts during the engineering of monitoring and diagnosis rules by reusable rule libraries and rule related modeling patterns specified with the Semantic Forms extension of SMW. The correct structure of the monitoring rules is verified during the engineering phase to satisfy RQ7 entirely. This is accomplished by an OWL Reasoner which processes the resulting monitoring ontology and executes specified consistency checks. On the one hand, we can define consistency checks based on metamodel structures, e.g. by verifying if the instances are consistent to their classes, and on the other hand, plant related consistency checks can be introduced, e.g. to verify if signal interfaces are linked in a correct way (e.g. to verify if "in" ports are correctly wired to "out" ports). Additionally, the engineering experts can use the concepts of the state recognition model to manually construct taxonomies of states, monitoring conditions or rules and to execute specified rule maintenance tasks by means of OWL reasoning which allows for meeting RQ9 wholly.

Approach R.f. A common rule format of the semantic web community that was compared with the solution proposed in this thesis, is SWRL. RQ1 and RQ3 can be satisfied entirely since SWRL relies on the same application-independent knowledge representation language as the plant models and the knowledge of these models can thus seamlessly be integrated in the rules. However, an important limitation preventing fulfillment of RQ2 is that it does not allow for rule reification due to a clear-cut separation between the terminological and the assertional level. Furthermore, RQ4 and RQ5 can only be met partially since SWRL supports only a limited set of algorithmic and logical operators and no temporal dependencies between dynamic plant engineering data can be defined. RQ6 and

RQ7 cannot be completely fulfilled due to a lack of rule libraries or templates for defining rule patterns or consistency checks based on them. Due to the missing reification ability, there is also no maintenance mechanism for semantically labeling or structuring of rules and RQ9 can thus not be satisfied.

Approach R.g. Another possibility to define monitoring rules is to hard-code them in Java which has no restrictions regarding applications and meets RQ1. Eclipse was used as generic editor for defining Java rules. Similarly to SWRL, there is no predefined adaptable metamodel for monitoring rules currently available which hinders fulfillment of RQ2. Java allows to satisfy RQ3 completely by dedicated APIs that process the plant knowledge stored in SMW and integrate the knowledge in the monitoring and diagnosis rules. Further, RQ4 and RQ5 are fulfilled by using Java libraries for implementing various algorithmic, logical and time sequence operators. Nevertheless, RQ6 cannot be satisfied, because there are no predefined libraries of temporal operators or rule-related modeling patterns available and such extensions have to be added manually to the best of our knowledge. Simple object-based consistency checks can be defined in Java, but they need to be hard-coded and are not part of the rule model. Thus, RQ6 is met only partially. RQ9 is satisfied in part by the ability of Java to provide maintenance information about rules, however advanced maintenance inferences considering the similarity of monitoring conditions cannot be established automatically.

Approach R.h. A complex event processing rule engine which could be used for implementing the SRM is *Etalis* which can be configured with the SWI-Prolog editor. Etalis satisfies RQ1 since it is application-independent. A drawback of Etalis is that its reification functionality is hard-coded. RQ2 can thus not be fulfilled. In consequence, mechanisms for semantically labeling or structuring rules are not available which hinders fulfillment of RQ9. Further, RQ3 is not met because plant knowledge cannot be integrated in the rules due to missing connections between Prolog and the TripleStore endpoint of SMW. RQ4 is only partly fulfilled since programming of algorithmic operations is possible, but cumbersome. Various kinds of time and logical operators as well as event processing operators are supported by Etalis which satisfies RQ5 entirely. Etalis is not an object-oriented programming approach, which means that specific rule libraries, templates or consistency checks cannot be defined. Hereby, Etalis cannot fulfill RQ6 & RQ7.

		RQ	***R.a* Rule Ontology + SMW + Drools**	***R.f* SWRL + Protege editor**	***R.g* Java + Eclipse editor**	***R.h* Etalis + Prolog editor**
CORE RQs		RQ1 (app. independence)	no restriction regarding application	no restriction regarding application	no restriction regarding application	no restriction regarding application
		RQ2 (reconfigurability)	reification functionality is supported and can be adapted	missing reification functionality, thus no adaptability	missing reification functionality, thus no adaptability	reification functionality is hard coded, thus no adaptability
		RQ3 (integration)	seamless integration since state recognition model relies on the same basis than plant models	relying on the same basis as plant models, their knowledge can seamlessly be integrated	knowledge of the plant models can be processed by dedicated APIs such as Jena	no connection between Prolog and SPARQL endpoint of SMW available
OUTER CORE RQs	STATE REC.	RQ4 (algorithmic & logical operations)	algorithmic and logical operators can be defined	only limited set of algorithmic and logical operators	broad set of algorithmic and logical operators provided	algorithmic operations are cumbersome, but logical operators are available
		RQ5 (time series analysis)	time series analysis by means of temporal operators is possible	temporal dependencies between dynamic plant data cannot be defined	no predefined libraries of temporal operators, extensions need to be added manually	event processing operators are available, e.g. SEQ, DURING
	USER GUIDANCE	RQ6 (modeling patterns & libraries)	reusable libraries for rules and rule related modeling patterns are available	libraries can be reused, but templates for modeling patterns do not exist	rule related modeling patterns and libraries could be established, but do not exist	no object-oriented programming approach, thus no templates or libraries
		RQ7 (consistency checks)	object-based and comprehensive plant related consistency checks possible	monitoring rules without constraint definition, thus no consistency check possible	hard-coded checks, which are not automatically part of the model	neither object-based nor plant related consistency checks available
		RQ9 (maintenance of rules)	state recognition ontology for semantically labeling and structuring of rules	due to missing reification, no mechanism for semantically labeling or structuring of rules	no advanced maintenance inferences considering the similarity of monitoring conditions	due to missing reification, no mechanism for semantically labeling or structuring of rules

Table 13.2.: Evaluation of approaches for the SRM (gray cells = RQ not fulfilled, blue cells = RQ partially fulfilled, white cells = RQ entirely fulfilled)

13.3. Summary of Evaluation

Summing up, our approach for the PMR *(M.a)* and the SRM *(R.a)* is the only implementation variants that meets all requirements. The only alternative that fulfills the essential core requirements concerning the PMR is Approach *M.b*. However, the usability requirements cannot be fully met by most of the approaches, except by our approach. This is mainly due to the ability of Semantic Web technologies to support the engineering experts in their modeling tasks by permitting experts to express queries and rules, define taxonomies and modeling patterns on higher levels of abstraction and navigate within complex data sets. The state recognition functionality requirements can only be entirely satisfied by Approach *R.a* which uses Drools, a powerful CEP rule engine. Based on our evaluation, plant experts can choose an appropriate approach for implementing the PMR and the SMR depending on their specific requirements, e.g. if extensive user guidance or advanced state recognition functionality is important, than Approach *M.a* is appropriate for implementing the PMR and Approach *R.a* for deployment of the SRM.

14. Evaluation of Usability

The previous chapter showed that the RMS requirements can be fulfilled by the approach specified within this thesis. However, to guarantee that the requirements address realistic usability issues of experts in the domain, a second aspect of RQ4, the 'usability' of the implementation proposed in Chapter 12, has to be tested:

> **Research Question 4:** What tools and procedures can be used to meet the RMS requirements and to improve ease-of-use through knowledge-based techniques to support plant engineering experts during the engineering process of a Resource Monitoring System?

During the engineering of the RMS, the users interact mainly with the plant model repository to insert plant knowledge on different modeling levels. The main task of the PMR is to guide the user in engineering and maintaining plant knowledge and monitoring rules. An assessment of the usability of the provided PMR is thus required by answering the following evaluation question:

> **Evaluation Question 4b:** Does the implemented plant model repository provide user guidance during engineering and maintenance tasks?

To test whether the proposed approach fulfills the needs of the target users, it is not implicitly necessary to have a huge group of test users. As pointed out in [150], the more severe usability problems are typically detected by the first few participants, and 80% of all usability problems are detected with only 4 or 5 participants. An important condition is however that the group of test users is representative and covers the target population.

To obtain necessary measures, experiments were conducted with four participants in the roles of knowledge engineers, plant engineers and plant operators with different levels of expertise. The participants were asked to fill out a questionnaire about their experience with the PMR after executing several tasks. With this questionnaire, the usability criteria *understandability, learnability, and operability* of standard ISO/IEC9126-1[1] were evaluated. Appendix A gives the questions, while the answers to the questions of the questionnaire are provided in Appendix A.2.

[1] ISO/IEC 9126-1, "Software engineering - Product quality," 2002.

In the beginning of the experiment, the participants got a short introduction on ontologies, UML, the conceptual modeling approach and Semantic Mediawiki, including a small test case to let them become acquainted with Semantic Mediawiki and the conceptual modeling approach. Then, they had to complete several tasks on different modeling levels.

14.1. Evaluation of Questionnaire

All outcomes of the evaluation of the questionnaire are discussed below and they are based on the answers given by the users during or after the experiment.

The users had different experiences in using Semantic Web ontologies and only one expert participant was tested. The previous experience of the participants with UML was moderate, while two of the users where beginners in modeling industrial plants, and one was an expert. Participants reported that they had only little tool experience with Protege and were more experienced with CAEX and SysML. Hence, the participants were actually more experienced in using generic software modeling tools and standards for industrial plant modeling, such as SysML or CAEX.

14.1.1. Users on Level M2

The task on level M2 was to define modeling entities and relationships of partial plant models (e.g. service model, product model, context model) by using predefined templates for “Entity Metatypes” and respectively “Relationship Metatype” as specified by the concept model. Considering questions on operability, three of the participants specified the difficulties of accomplishing the tasks as “low”, whereas the fourth participant answered “average”. During the experiment, it became clear that participants without any experience in OWL found it more difficult to accomplish the tasks. The tool support provided by predefined modeling templates and forms, consistency checks, namespaces and browsing ability was considered as “good” by three participants and “inadequate” by a user which is not experienced with OWL and SysML. Furthermore, the user support by means of consistency checks and queries was rated with “rather yes” by three participants and “absolutely” by one participant. Main obstacles found during the task were a “lack of visualization of plant knowledge”, the “learning curve” and the difficulty of “getting used to the complexity of the task”. This obstacles could be improved in the future by installing an additional visualization extension proposed by Semantic Mediawiki and by providing more guidance to the users when modeling different kinds of plant knowledge.

Answers on the learnability questions showed that an average time of 6,75 working hours (almost one average working day) was needed to get acquainted with the conceptual modeling approach, while the time for learning how to use the concepts of the concept model was approximately 10 hours. The time for learning how to use this approach and tool is thus within an acceptable time frame of two working days.

Questions on the understandability of the approach and the tool showed that explications and examples in the Semantic Mediawiki were considered as helpful to understand the concepts. Further, navigation between different levels of abstraction was considered less intuitive than navigation between different plant aspects. This is probably due to the fact that most of the users are not experienced in using metamodeling approaches where type and instance specification play an important role. Three of the users considered the proposed conceptual modeling approach as “absolutely” useful for modeling different aspects of industrial plants. The required actions for creating modeling entities were found “rather” clear and intuitive. The evaluation of the understandability showed that even if navigation’s between entities and required actions are not “absolutely” intuitive, the usage of the proposed approach is considered as very useful for modeling different plant aspects.

14.1.2. Users on Level M0 & M1

In a second study, the same participants were asked to accomplish tasks of user roles on level M0 and M1. The tasks for user was first to define a new component type, then to instantiate this component type and finally to exchange an existing component instance. Considering questions on operability, the participants specified the difficulties of accomplishing the task 1 as “low” or “average”, whereas the third task was considered by 3 participants as “average”. The tool support provided by predefined modeling templates and forms, consistency checks, namespaces and browsing ability was considered as “good” by all participants. Furthermore, the user support by means of consistency checks and queries was rated with “rather yes” by two participants, and “not really” and “absolutely” by respectively one participant. Main obstacles found during the task were similar to those on level M2, only the “attribute heritage” was found as additional obstacle. This obstacle is due to the difficulty to understand how to use the rules for automatically inherit attributes from types to instances within SMW.

Answers on the learnability questions showed better results than for level M2. An average time of 4,5 working hours was needed to get acquainted with the conceptual modeling approach, while the time for learning how to use the concepts of the concept model was approximately 10 hours. The time for learning how to use this approach and tool is thus within an acceptable time frame.

Questions on the understandability of the approach and the tool showed that navigation between different levels of abstraction, e.g. from motor instance to a motor type, was considered less intuitive than navigation between different components. Usual modeling tools (e.g. SysML) handle type and instance levels in separate diagrams. Therefore, the navigation between these two types of levels is probably considered as less intuitive. Three of the users considered the proposed conceptual modeling approach as “absolutely” useful for modeling different aspects of industrial plants. The required actions for creating modeling entities was found as “rather” and “absolutely” clear and intuitive. The evaluation of the understandability showed that even if navigation’s between entities are not qualified as “absolutely” intuitive, the usage of the proposed approach appears to be very useful

for the modeling of different plant aspects on the M0 and M1 level, such as connections between plant components and relations from component instances to component types.

14.2. Summary of Evaluation

The results of the questionnaire with respect to the three criteria of usability are summarized in Table 14.1. The participants of the experiments confirmed the ease of use of the proposed systems, reported that the learnability is within an acceptable time frame and stated that the understandability of the conceptual modeling approach is rather intuitive. Nevertheless, a certain understanding of concepts of Semantic Web and object-oriented paradigms is necessary for learning how to use concepts introduced in the concept model. Furthermore, these results gave first hints on how to optimize the usability, such that visualization extensions and additional explanations were subsequently added to SMW.

Criteria & Definition	Results of Evaluation
Operability: Ability of the software to be easily handled by a given user	Most of the participants found it easy to handle the tools and reported that they were supported by special features of SMW, e.g. modeling templates and forms. The only drawback was that participants who are not common to semantic web or object oriented paradigms needed additional support in understanding basic concepts of the concept model, especially the usage of triple store and component types required more explanations.
Learnability: Evaluates learning efforts for different users	The time for learning how to use the conceptual modeling approach and tool was approximately 2 working days and is thus within an acceptable time frame.
Understandability: Determines the ease of which the functions of a system can be understood	Participants reported that navigation's between modeling levels was not very intuitive, nevertheless the proposed setting is considered as an appropriate, supportive tool for modeling different plant aspects on different levels of abstraction.

Table 14.1.: Evaluation of criteria of usability

Part V.

Application Scenarios

15. Application Scenarios for Resource Monitoring

In this Chapter, *application scenarios* are defined to evaluate and improve the design of the RMS. The basic idea of an application scenario (AS) is a "story told from the point of view of one or several people who want to achieve a given result" as specified by [7]. In this chapter, a story on how the RMS defined within this thesis can be embedded in a realistic manufacturing context is told. To illustrate the scenarios in a demonstrative manner, three real industrial plants were used to demonstrate different AS. All resource monitoring aspects considered throughout the last chapters are summarized and discussed within these ASs.

Towards this end the ASs demonstrate the application of the RMS on different levels of analysis as described in Section 15.1. Multiple scenarios were implemented on a Siemens Conveyor Plant as presented in Section 15.2. Section 15.3 illustrates the Smart Key Finder Plant of DFKI where a context-dependent AS was realized by means of a context broker. Finally, Chapter 16 evaluates the described ASs based on different criteria of performance and Chapter 17 evaluates the reconfigurability of the provided approach.

15.1. Levels of Analysis

To fully understand the resource usage of industrial plants, these plants need to be studied at different levels of analysis. These levels of analysis correspond to the functional plant hierarchy specified in Section 2.2.1 and range from the enterprise level (level 4), where the rough production planning is administrated, to the field level (level 0), where information from the technological process through sensors is collected and processed as specified in more detail in standard IEC 62264-3[1].

The levels are characterized by different temporal scales ranging from days at the enterprise level to micro-seconds at the field level. An RMS should allow to make decisions across multiple levels of analysis. Individual plant components, such as field devices, PLCs, Human Machine Interfaces (HMIs), and control systems, are specifically attributed to individual levels.

The ASs presented in the subsequent sections demonstrate the application of the resource monitoring framework using resource usage and process parameters from ma-

[1] IEC 62264-3 (2007): "IEC 62264-3: Enterprise-control system integration – Part 3: Activity models of manufacturing operations management"

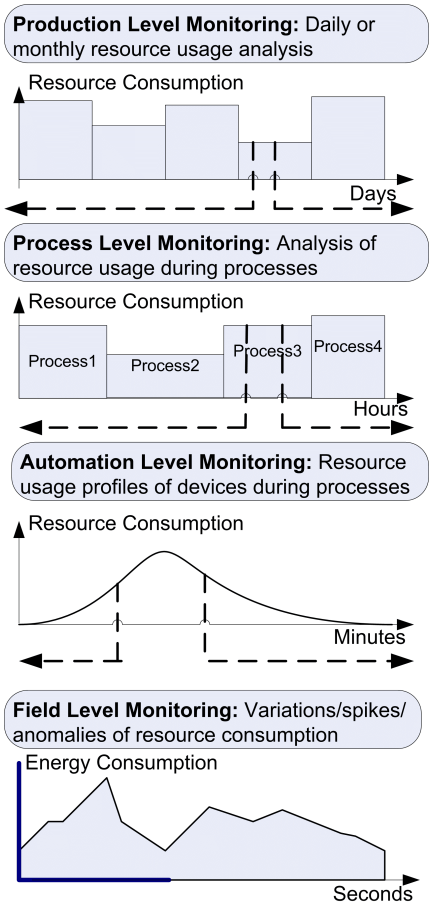


Figure 15.1.: Resource Monitoring on different levels of analysis

chining experiments on all levels of the automation pyramid except the enterprise level. Figure 15.1 shows types of monitoring analysis required across some of these levels. The highest level of analysis is the *Production Level Monitoring* where the resource usage is analyzed based on a daily or monthly basis and reported to the plant owner to improve the plant processes on a long-term basis. On the *Process Level*, resource monitoring is needed to observe the actual resource usage within a production process by comparing it to the target resource usage. Resource usage profiles are further verified to monitor components on the Automation Level. And finally, the detailed anomalies in the resource data are detected and reported on the Field Level of the plant.

The first precondition to support analysis across different levels is a modular architecture of the RMS, which is given by its decentralized character as described in Chapter 12. As shown in [148] for energy data, there is further the need to analyze the temporal aspects of the measured data in order to place it in the context of the manufacturing activities.

The automated RMS presented in the previous chapters allows to attach such contextual process-related information to the raw data to address temporal monitoring aspects on all levels of analysis.

Depending on the time scale and the typical plant components (MES, HMI or PLC) involved in the AS, the according AS can be assigned to different levels of analysis. An example is the prediction of energy threshold as implemented in AS3. Either the prediction warning is reported to the plant owner based on a daily or hourly threshold, or in case of complex, energy-intensive processes, the operator on duty is alarmed if the energy consumption of a certain production process is approaching its upper limit. This AS can thus be implemented on Production Control or Process Control Level.

15.2. Siemens Conveyor Plant

Several application scenarios (ASs) were implemented on a test plant at Siemens CT built up in the scope of the RES-COM project. This test plant is named “Siemens Conveyor Plant” since it transports objects on a conveyor belt and measures the charging levels of the transported objects. The front side of the plant with its principal components is depicted in Figure 15.2. The main components of the plant are the **HMI Panel** for visualizing the ongoing process, a **Profi Energy** connection for the external communication, an **Energy Controller** unit to control the energy consumption within different standby levels, a **Controller** to control the production process, a **Motor Drive** to control the motor speed and a **Sentron** to measure the input power of the components. To understand the structure of the resource monitoring ASs, the process executed by the plant and principal components are explained subsequently in more detail.

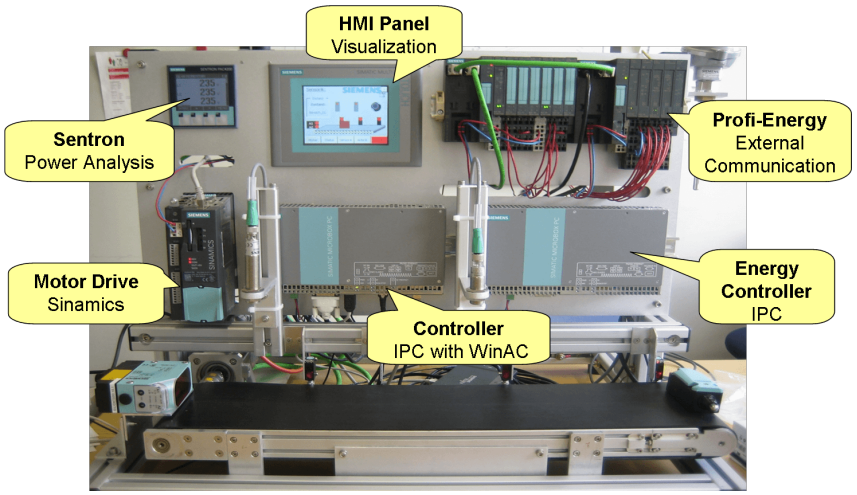


Figure 15.2.: Front view of Conveyor plant demo

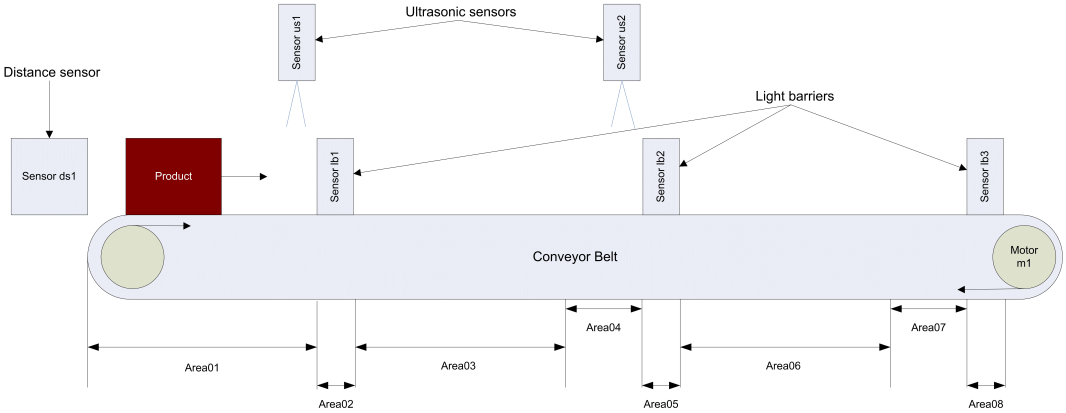


Figure 15.3.: Schematic view on areas of the Conveyor plant

Activity	**Description**
Init	Plant components are initialized
Ready	Plant is set to operating mode and waiting for an object
MoveToArea (02/05/08)	Object on the conveyor belt is moved to area 02, 05 or 08 respectively
Measure (1/2)	Filling level of the object is measured by one of three different sensor s1 or s2
DecelerateConv (1/2)	Conveyor belt is decelerated within this activity

Table 15.1.: Description of Activities shown in Figure 15.4

The conveyor belt is divided in different areas as presented in Figure 15.3. *Area02*, *Area05* and *Area08* are kept under surveillance by three light barriers of type *Simatic PXO300*. *Area01* and *Area04* are equipped with ultrasonic sensors of type *SIMATIC PXS200* for measuring the charging level of objects. An additional distance sensor of type *SIMATIC PXO650* mounted at one side of the conveyor belt is further constantly checking if an object is positioned on the conveyor belt.

The production process executed by the plant starts when an object is placed at *Area01* on the belt and the plant is residing in the state *Ready*. This object is then transported to *Area02* to measure its higher level charging level and display the result of the measurement on the HMI panel. In a next step, the object is transported to *Area05* where a lower level measurement is executed and transported to the end of the conveyor belt (Area08). While the object is transported with constant speed within *Area03* and *Area06*, is has to be decelerated within *Area04* and Area07. The entire production process is depicted with links to the HW component templates participating in the activities in Figure 15.4 as UML diagram. A short description of the respective activities is given in Table 15.1.

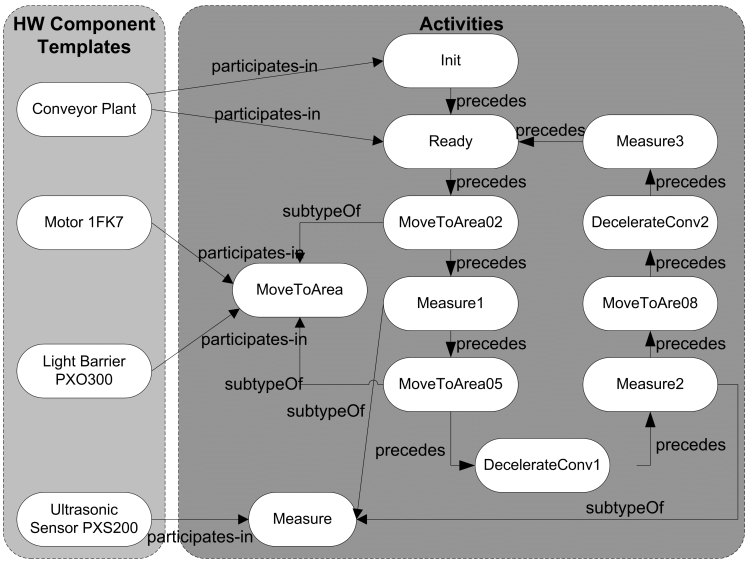


Figure 15.4.: Relations between activities and HW component templates of the conveyor plant

The motor mounted in the plant is a servo motor of type 1FK7. To simulate engine wear or blockades affecting the movement of the conveyor belt, an additional motor break was installed. This motor break can be activated manually by the plant operator at any time and increases the motor torque.

In the subsequent sections, the application scenarios for resource monitoring realized on the Conveyor Plant Demo are presented and discussed in more detail.

15.2.1. AS1: Standby Power Monitoring

The reduction of energy demand during standby mode of industrial plants contributes to the overall energy efficiency of the plants. A mathematical modeling approach specified by [87] serves as basis in this AS for identifying optimal strategies analytically to quantify the energy savings potentials during standby mode and give support for technical realization. This approach is embedded on an *Energy Controller* to switch plant components off or on during standby mode by using a predefined *Energy Model*. However, constraints on sequence and time of these switches increase the complexity and failure rate of the system. Thus, the switchover between standby states has to be monitored to prevent such failures.

The purpose of this AS is to monitor the power consumption of the conveyor plant during standby mode. The RMS detects incorrect sequences of standby states by comparing the power consumption of the target standby state with the actual power consumption of the current state as presented in [88]. A diagram of the Resource Monitoring Demo in Figure 15.5 shows the target standby power specified for standby states in the Energy Model

of the Energy Controller in blue and the actual energy consumption recorded during the current standby state in red.

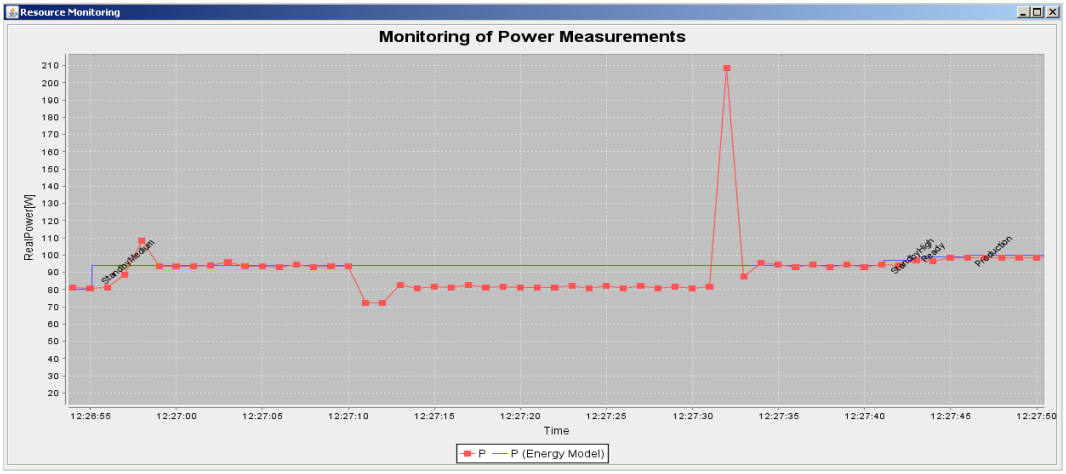


Figure 15.5.: Diagram of target standby power compared to actual standby power

To realize this AS, a SENTRON PAC 4200 power meter was mounted at the conveyor plant to measure the power consumption of the plant components. The sentron can only measure the power consumption of one electrical circuit. Several electrical circuits of the conveyor plant including multiple components as depicted in Figure 15.6 need to be monitored. Structural knowledge incorporated in the structure model is needed to identify components, which are not switched into standby mode correctly. Therefore, the target power consumption of the plant components is stored in the structure model in the component attribute “standbyPowerConsumption”. When the target power consumption is exceeded within an electrical circuit, then the state “standbyPowerExceedance” has to be computed by the monitoring system depending on the current standby state.

The AS was implemented for electrical circuit *ec1* of *sentron1*. Several rules need to be implemented for the different standby processes. For example, a rule for the process “standbyMedium” is implemented that states “if the expected standby power consumption of *ec1* exceeds the expected power consumption in process *standbyMedium* by *diffStandbyPower*, which corresponds exactly to the standby power of a component *comp1*, which is part of *ec1*, then the state of the component is set to *standbyPowerTooHigh*”. The rule has the format:

```
RULE standbyMediumPowerExceedance:
   resides-in(standbyMediumPowerTooHigh,?comp,Time1) ←
   executes(plant1,standbyMedium), measured-value(sentron1,?standbyPowerEc),
   maxStandbyMediumPower(?ec, ?maxPowerEc),
   isBiggerThan(?standbyPowerEc, ?maxPowerEc), hasPart(?ec,?comp),
```

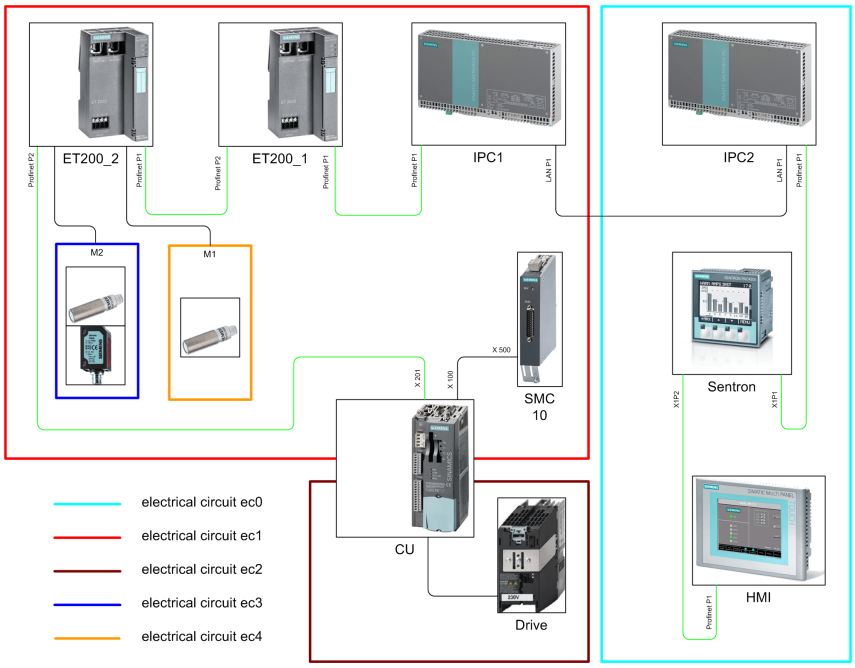


Figure 15.6.: Schematic overview on electrical circuits in the conveyor plant

```
    standbyPower(?comp,?diffStandbyPower),
    ?diffStandbyPower = ?maxPowerEc - ?standbyPowerEc.
```

The total amount of required rules for this AS depends on the amount of standby states of the plant. For every standby state three rules are needed to determine if the standby power consumption is either too high, too low or in the normal range. In the case of the conveyor plant, three standby states are possible, which means that 9 rules are needed for this AS.

15.2.2. AS2: Component Wear Monitoring

Despite high-end materials and refining procedures, components in industrial plants are still subject to wear due to deterioration or erosion. Component wear is an important resource within the plant environment since a high wear decreases the life time of components significantly. Current research focuses more on analyzing the reasons for this component wear in the manufacturing environment than on monitoring the wear of components. For example, the effects of engine operating variables and the quality of the lubricating oil on the wear of engine components was examined in different tests by [133].

Temperature changes or sudden load switches due to switching operations increase the component wear and influence the component life time. The failure rate of components increases with the number of switching operations during production processes. The con-

tinuous registration of the number and frequency of switching operations applied on components allows for providing wear analysis results and sending alarms prior to a failure. The plant operators can thus optimize processes to reduce the component wear within affected process steps. Such an optimization raises scheduled maintenance and exchange intervals of components and allows plant operators to interrupt the production process during non-critical steps to exchange components prior to their breakdown.

In this AS, the failure rate of components of the conveyor plant due to switching operations is considered. Different factors need to be taken into account to compute the failure rate. These factors include the switching capacity, the switching frequency, and the type of load (ohmic, inductive or capacitive). The formula to compute the failure rate based on these factors is $\lambda = \frac{0,1C}{B10}$. In this formula, λ is defined as the failure rate per hour; C is defined as switching cycles per hour and $B10$ encompasses the statistical amount of switching cycles causing 10% of the components to brake.

The Power Module ET200S and the SIRUS Hilfsschütz of the conveyor plant are used as an example to demonstrate this AS since the B10 value is explicitly specified in the data sheet of their manufacturer. The B10 value depends on the switching cycles of the Power Module in different time frames as shown in Table 15.2. Within the current plant configuration, a B10 value of 1.100.000 switching cycles is used for the Power Module.

Switching cycles	Time Frame	B10	λ
1		1.000.000	$4,17x10^{-9}$
10	24 hours	1.000.000	$4,17x10^{-8}$
100		1.000.000	$4,17x10^{-7}$
1		1.000.000	$1,00x10^{-7}$
10	1 hour	1.000.000	$1,00x10^{-6}$
100		1.000.000	$1,00x10^{-5}$
1		1.000.000	$6,00x10^{-6}$
10	1 minute	1.000.000	$6,00x10^{-5}$
100		1.000.000	$6,00x10^{-4}$

Table 15.2.: Influence of switching cycles on failure rate, source [125]

The amount of switching cycles is constantly registered by the control unit. Based on this input, the Monitoring Unit computes the state of components, which can exceed the failure rate limits in two cases: 1) the amount of on off switches in a certain time frame is too high, 2) the maximum amount of switching cycles in the life time of the component is reached. In these cases, the Monitoring Unit computes the monitoring state “FaultRateTooHigh” and “maxFaultRateExceedance” respectively. Thus, two rules are needed to determine if the fault rate is too high or in the normal range. An exemplary rule has the following format:

```
RULE maxFaultRateExceedance:
   resides-in(maxFaultRateExceeded,?comp,Time1) ←
```

```
measured-value(counter1,?faultRate), maxFaultRate(?comp, ?maxFaultRate),
isBiggerThan(?faultRate, ?maxFaultRate).
```

15.2.3. AS3: Threshold Prediction

The notion of *Demand Side Management (DSM)*, also known as *Energy Demand Management* was first introduced by the Electric Power Research Institute in the 1980s [93]. The goal of DSM is to encourage consumers to use electricity in a uniform way, or to move the time of energy use to off-peak times such as nighttime. The energy suppliers are charging less money for a uniform electricity usage of consumers and more for electricity usage peaks. The DSM does not reduce total energy consumption, but is expected to reduce the need for better supply networks and fuel cost and to increase the efficiency of system investment as stated in [136]. Thus, this method allows to reduce resources in terms of financial investigations and natural resources needed by energy suppliers [103].

To allow for customers of energy suppliers to avoid peaks in their electricity usage, a specific energy limit has to be specified for a limited time frame, which varies depending on peak and off-peak times. The aim of this AS is to support the customers in respecting this limit by predicting the point in time when the plant will exceed this energy limit. In consequence, the RMS warns the plant operator a certain time in advance so that the plant configuration and processes can be changed accordingly (e.g. by switching from full load to half load mode).

To demonstrate this scenario, the energy usage of the entire conveyor demo plant was considered as monitoring parameter. Based on results of multiple energy measurements, a linear regression model was defined to describe the energy usage profile of the plant for this scenario as shown in Equation 15.1. Based on this model, the expected energy value of a monitored item at a certain time can be computed. The coefficients of the linear regression model were computed by means of Equation 15.2 . These coefficients are shown in Figure 15.7 and involve the prediction time, the energy usage threshold and the time frame in which the energy usage is accounted. Prior to an energy exceedance, the Monitoring Unit of the RMS computes a Monitoring State of the plant including the time to exceedance and the current energy usage value to allow the plant operators for acting accordingly. The linear regression model is computed by the controller and the result can be processed by a monitoring rule. To this end, only 2 monitoring rules are needed for this AS, one rule to compute the failure state and one rule to compute the normal state.

$$y = a + bx \tag{15.1}$$

$$\sum_{i=1}^{n} (y_i - (a + bx_i))^2 \rightarrow min \tag{15.2}$$

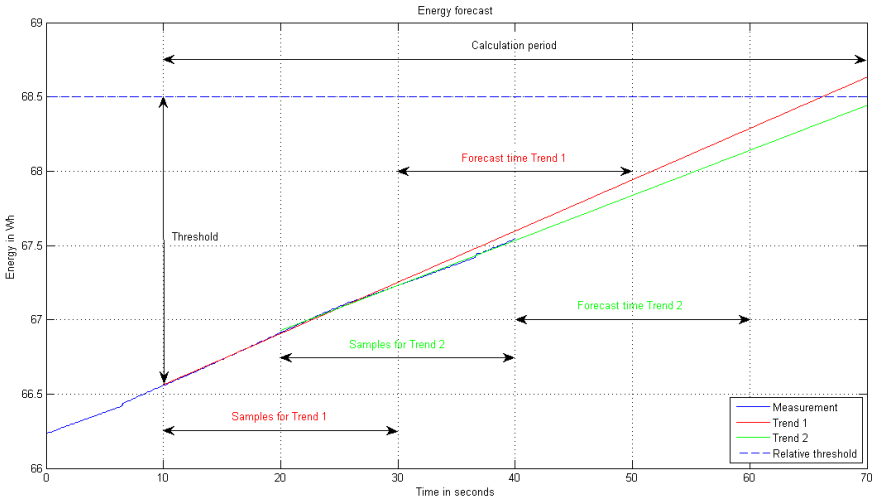


Figure 15.7.: Energy diagram to determine values for linear regression model

15.2.4. AS4: Activity Sequence Monitoring

Complex products require several complex production processes. Increased total resource consumption for successive products may indicate inconsistencies or process plan deviations. To identify such indicators by analyzing the resource usage of plant processes, temporal aspects need to be considered and placed in context of the activities executed by the components [148]. To this end, the parameters *sequence*, *duration* and *resource usage* of an average or target activity have to be registered once at plant implementation. Specific activities of the production process can be analyzed by comparing them to this average or target activities. Deviations or inconsistencies in the activities can thus be detected by the RMS. This allows to identify and remove scrap or waste products at an early stage of the entire production process. The quality of the production processes can be optimized based on the monitoring results, which reduces rework costs, the amount of waste products and the total resource usage.

This AS has to be realized by means of time series analysis. An effective means to model and analyze time series are event stream processing techniques, such as Complex Event Processing (CEP). These techniques are used in this AS to create abstract events and reason on them by pattern identification and matching. A powerful algorithm of CEP systems is the Rete algorithm, which provides an efficient way of pattern matching as specified in [48]. The open source rule engine *JBoss Drools* used for reasoning in the conveyor plant is based on this algorithm.

To illustrate this AS, an ideal target production process of the conveyor plant was once

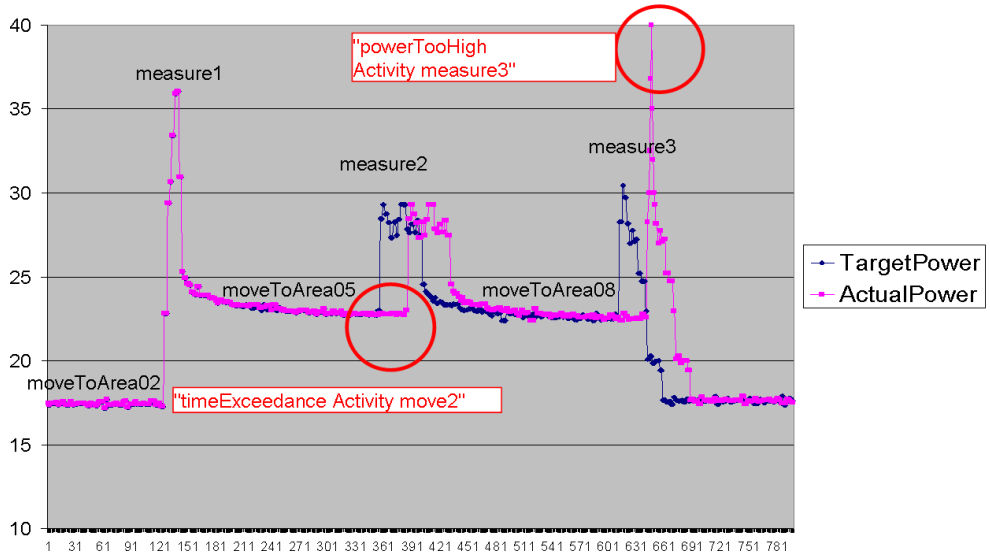


Figure 15.8.: Comparison of target and actual power consumption during the production process of the conveyor plant

registered and then compared to an actual process. Am example for such a comparison of target and actual process is presented in Figure 15.8 that shows two different deviations detected by the RMS:

1. the activity *moveToArea02* exceeds the time limit → state "timeExceedance Activity move2" is computed by the RMS
2. high peak is detected by the RMS during activity *measure3* → state "powerTooHigh Activity measure3" is computed by the RMS

The rule implemented to detect a time exceedance of activities for this AS is:

```
RULE timeExceedanceActivity:
   resides-in(activityTimeExceedance,?activity, ?actualDur,?targetDur) ←
   executes(plant1,?activity), measureValue(?activity, ?actualDur),
   participatesIn(plant1,?activity), durationActivity(?activity,?targetDur),
   isBiggerThan(?actualDur, ?targetDur).
```

Other possible faults that can be detected by time series analysis are: 1) an activity in the process sequence is skipped, 2) the product is lifted from the conveyor belt, 3) the motor brake increases the energy consumption needed to execute an activity. Depending on the kind of fault, the MU computes different monitoring states such as “Error Wrong Activity Sequence” and displays the expected values and the actual values. An additional rule is

needed to indicate the normal state. Thus, the total amount of rules needed for this AS is 4. To specify a rule that verifies the correct sequence of activities, a special CEP operator is needed. This operator is named “seq” and indicates a sequence of two events. Further, the relationships *precedes* of the PSL extension Occurrence Tree stored in the process model is needed. The binary predicate *precedes(o1, o2)* denotes an ordering relation over activity occurrences, so that they form trees. The following rule was specified to detect wrong sequences of activities:

```
RULE activitySequenceDeviation:
   resides-in(wrongActivitySequence,?activity,?targetActivity) ←
   executes(plant1,?currentActivity), seq(?lastActivity,?currentActivity),
   ¬ precedes(?currentActivity, ?lastActivity).
```

15.2.5. AS5: Energy Efficiency Monitoring

The evaluation of energy efficiency of manufacturing processes require a clear semantic description of related automation systems as shown in [33]. The ontology proposed there can be used for implementing an energy efficiency monitoring scenario. In this scenario, the degree of efficiency of components is considered in dependance on their optimal operating point. The highest degree of efficiency of components can be reached at an optimal operating point characterized by specific parameters. An example is the energy efficiency of electrical motors, which depends on the rotational speed of the motor. The motor can reach maximum energy efficiency at a predefined optimal rotational speed specified by the manufacturer. The energy efficiency η of a motor is defined in general terms as:

$$\eta = \frac{P_{mech}}{P_{el}} = \frac{M \times n}{U \times I} \; (M = \textit{torque}, n = \textit{rotational speed}, U = \textit{tension}, I = \textit{electric current})$$

For example, the Siemens Motor1FK7 reaches its optimum operation point at a rotational speed of 6000 RPM as presented in Figure 15.9. By means of a continuous RMS, the energy efficiency of all plant components can be constantly analyzed and displayed to identify the ideal load conditions of the plant. This information allows the plant operators to optimize the processes and load conditions to maximize the energy efficiency of the plant. Over time, deviations from this maximum energy efficiency of the plant due to component wear and erosion can also be detected. Support in deciding if a component has to be exchanged from a resource efficiency perspective is thus given by the RMS.

The component Siemens Motor 1FK7 is used in the conveyor demo plant to demonstrate this AS. A normal profile of the energy efficiency of Motor 1FK7 when executing a standard production process and the same production process with an active motor brake is presented in Figure 15.10. The energy efficiency is almost half of the expected efficiency when using the brake. This application scenario is also helpful to identify general deviations due to heavy-weight objects on the conveyor belt or wear in the motor bearing. A simple threshold rule is needed for this scenario to determine if the energy efficiency

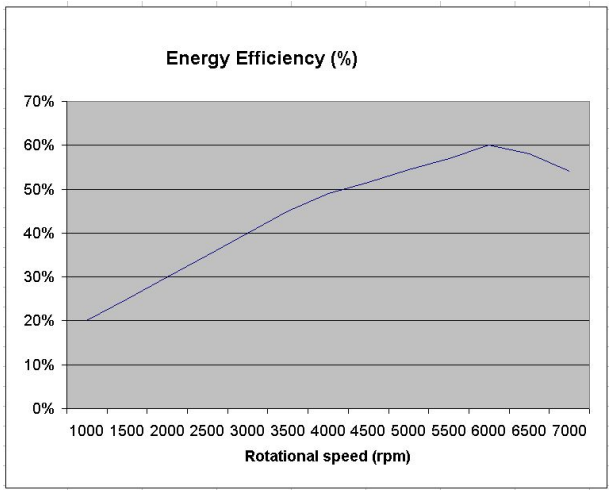


Figure 15.9.: Energy efficiency of Siemens Motor 1FK7 in dependance of its rotational speed

is lower than a predefined threshold. Another rule is needed to determine if the energy efficiency is within normal range.

15.3. DFKI Smart Key Finder Plant

In the context of the RES-COM project, the RMS was implemented on a plant of DFKI[2], which produces smart key finders as shown in Figure 15.11. To demonstrate the resource monitoring approach, the transportation block of the plant was used as representative example.

The transportation block of the plant is used to transport smart key finders on two conveyors *cv1* and *cv2*, which are both running with same constant speed. The two conveyors are driven by the motors *m1* and *m2* respectively. These engines include power modules *pm1*/*pm2* to vary their rotational speed controlled by control units *cu1*/*cu2*. The entire transportation unit is grouped in the functional drive group *g* as shown in Figure 15.12.

In the next Section, an AS for resource monitoring as implemented on the smart key finder plant is presented and discussed in more detail.

15.3.1. AS6: Context Monitoring

A monitoring system can make more intelligent decisions and determine more accurately the situation of the monitored item if additional information (=Context) is provided about all entities that are in some way relevant to the current monitoring task. Furthermore, this

[2]German Research Center of Artificial Intelligence

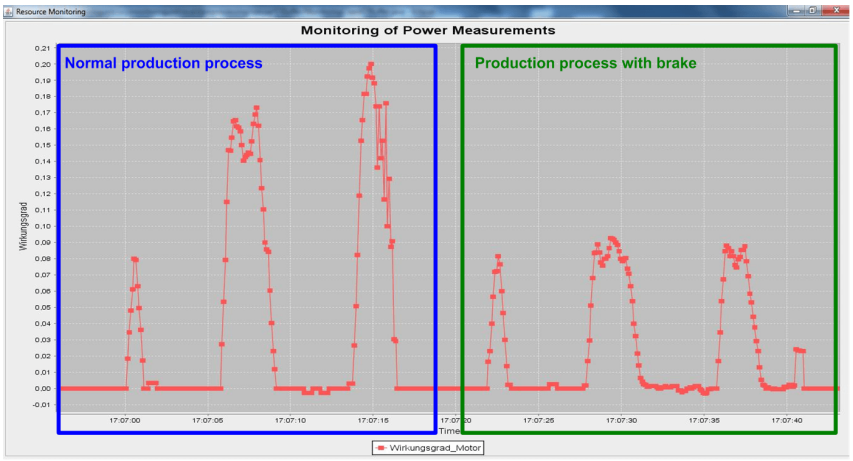


Figure 15.10.: Diagram of energy efficiency of Motor 1FK7 in normal condition and with motor brake

context information is an important background knowledge that can be presented to the plant operator to allow him to optimize the plant considering the current context.

A special feature of the Smart Key Finder plant is that the operator of the plant can choose between two production contexts: a) produce with low energy consumption, b) produce in minimum delivery time. The MUs in the plant are supplied with context information from a central server, the *context broker*, that collects the context information from all participating sources in the plant. The context broker supports several protocols (e.g. OPC UA, Web services, REST) to call the technical interface of the ADPM, read the enabled variables and store the values in its internal database. The individual MUs can request and register all context information that is relevant to their specific monitoring task according to their underlying context and structure model.

If the operator chooses the first context option (*low energy*), then the engines rotational speed is adapted to reach the maximum energy efficiency and the two conveyors run at a lower speed. If the operator chooses the second option (*min delivery time*), the engines rotational speed is adapted to its maximum to allow a fast transportation of the product by the conveyors. To verify if the control system is correctly adapting to the production contexts, several monitoring rules were defined within this AS. For instance, a rule that states “If the production context is *lowEnergy* and the total power consumption of group g exceeds a predefined *maxInputPower1*, then the group g1 resides in state *energyTooHigh*”. This rule has the following format:

```
RULE energyTooHighContextLowEnergy:
   residesIn(energyTooHighInContextLowEnergy,?comp) ←
   currentContext(?comp,lowEnergy), measured-value(?comp,?inputPower),
```

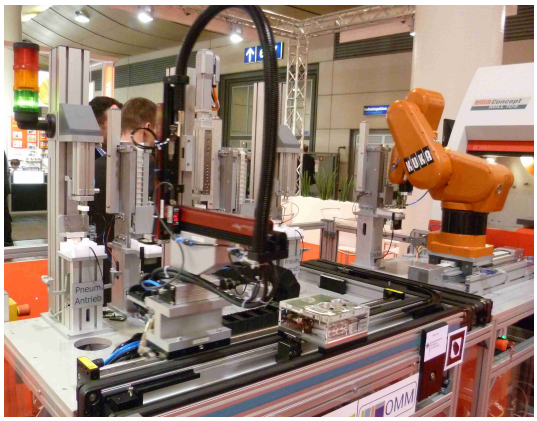


Figure 15.11.: Front view of smart key finder production plant

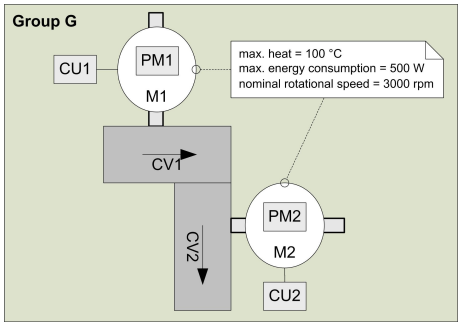


Figure 15.12.: Transportation scenario with conveyors *cv1, cv2* driven by engines *m1, m2* including power modules *pm1, pm2* and control units *cu1, cu2*

```
    maxinputPower1(?comp,?maxInputPower),
    biggerThan(?inputPower, ?maxInputPower).
```

Additionally, a rule was defined to computes a similar state, but under different conditions. This rule says “If the production context is *minDeliveryTime* and the total power consumption of group g exceeds a predefined *maxInputPower2*, then the group g1 resides in state *energyTooHigh*.

```
RULE energyLimitContextEnergyTooHigh:
   residesIn(energyTooHigh,?comp) ← currentContext(?comp,minDeliveryTime),
   measured-value(?comp,?inputPower), maxinputPower2(?comp,?maxInputPower),
   biggerThan(?inputPower, ?maxInputPower).
```

16. Evaluation of System Performance

Some AS are time critical and require a monitoring state to be computed in a specific time frame. For example, the time frame for computing the threshold prediction in AS3 has to be smaller than the specified prediction time. Thus, the system performance question formulated in research question RQ5 has to be answered:

> **Research Question 3:** What is the influence of an increase in different scalability factors on the scalability and performance of a resource monitoring system realized by means of knowledge-based technologies?

Various factors play important roles when evaluating the performance of a monitoring system. According to [143], the performance of a system is significantly influenced by "the amount of data, the frequency with which the data is transmitted, the speed of data transmission, latency, and the data transmission route". To evaluate the performance of the RMS implemented on the conveyor plant, the overall **computation time** of the system was calculated when modifying different scalability factors. The computation time considered for the performance evaluation is the time for computing a monitoring state based on monitoring input data. An evaluation question EQ5b was thus defined to evaluate this second aspect of RQ5:

> **Evaluation Question 3b:** To what degree is the computation time of the implemented resource monitoring system influenced by the sampling frequency or the amount of rules?

Two different performance factors are tested in the next Sections to answer EQ 5b. First, the system performance was tested at different sampling frequency, then the influence of an increasing number of rules on the computation time was analyzed.

16.1. Sampling Frequency

The default sampling rate of Simatic Net OPC UA Server implemented on the conveyor plant is 100 kHz. But for the detection of power peaks, a minimum sampling frequency of 10kHz is required as shown in [67]. A ring buffer was thus implemented on PLC side to guarantee a higher sampling frequency to detect all relevant power peaks. Tests were

conducted to measure the computation time of the RMS at different sampling frequencies. The results of the tests as presented in Table 16.1 demonstrate that the computation time of the RMS is lower than 1 ms for different sampling frequencies down to 5kHz. A computation time of 1 ms is sufficient for time critical ASs such as AS3 and thus within acceptable range.

	Sampling frequency			
	40 kHz	20 kHz	10 kHz	5 kHz
Computation time	<1ms	<1ms	<1ms	<1ms

Table 16.1.: Computation time at different sampling frequencies

16.2. Amount of Rules

[13] showed that an increase in the number of parts of rule-based systems leads to an increase in complexity, which lowers the system performance. Thus, the influence of the number of rules on the performance of the RMS has to be evaluated. The RMS uses the rule engine JBoss Drools for rule execution based on Rete 2 algorithm described in [107]. A special feature of Drools is that if a condition of a rule changes, *incremental score calculation* as specified in [66] will calculate the delta with the previous state to find the new state, instead of recalculating the entire state on every condition evaluation. The speedup due to this incremental score calculation is huge, because the speedup is relative to the size of the planning problem.

Tests were conducted to measure the computation time of the RMS implemented on the conveyor plant at increasing number of rules in the rule base. First, only one rule was inserted, then rules were added up to the maximum number of rules needed to realize all ASs. As depicted in the results in Table 16.2, the performance of the system was not decreasing with increasing amount of rules. This is mainly due to the efficient incremental score calculation of Drools, since the rules are only activated when monitoring states change. But in normal plant conditions, the states of all plant objects are not changing simultaneously. The usage of Drools allows thus to guarantee a high system performance at increasing plant sizes.

	Number of rules		
	1	5	19
Computation time	<1ms	<1ms	<1ms

Table 16.2.: Computation time at different number of rules

17. Evaluation of Reconfigurability

Several evolutions in manufacturing require a higher reconfigurability of monitoring systems as argued in Section 4. The subsequent chapter is dedicated to evaluating the reconfigurability of the RMS build on the monitoring model with the underlying conceptual modeling approach. This evaluation answers research question 5:

> **Research Question 5:** How can knowledge-based techniques be used to improve reconfigurability of resource monitoring systems?

Within this evaluation, the reconfigurability is tested, since it can be seen as a basis for a changeable factory according to [158]. As basic definition of this term, the meaning introduced by [44] is used which defines reconfigurability as the “ability to change the behavior of a system by changing its configuration”.

Systems supporting the two aspects *integration* and *reuse* allow a higher degree of reconfigurability. The monitoring model with its underlying conceptual modeling approach as presented in Chapter 8 establishes an explicit formal specification of concepts and their relations for the manufacturing domain similar to ontologies (following the definition of the term ontology by [56]). In general, ontologies are said to facilitate integration and reuse of valuable knowledge across applications [158]. This statement needs to be verified when using the RMS established within this thesis.

The most common reason for a system’s reconfiguration is that an installed component brakes and has to be exchanged by a new component. In this case, the monitoring system has to be adapted partially depending on the characteristics of the new component. The evaluation question used to evaluate RQ3 is thus defined as:

> **Evaluation Question 5:** What is the effort for modifications within the monitoring system if an installed monitored component in a plant is exchanged with a new component?

To measure the complexity of the modifications necessary to exchange a plant component in a given monitoring system, an exchange workflow has to be defined. This exchange workflow consists of several steps needed to exchange a component in a monitoring system. Every step has a certain complexity depending on the task that has to be accomplished within this step. The complexity of a step depends on the modifications needed.

To evaluate the amount of modifications, two different settings were tested to identify the exchange workflow for each setting. First, a usual monitoring system was tested and the steps to exchange a monitored component were specified (Exchange Workflow 1). The result was compared to Exchange Workflow 2 of a monitoring system based on the models established within this thesis consisting of the monitoring model specified in Section 10.2 and the conceptual modeling approach described in Section 8.1. A direct comparison of the two workflows is shown in Figure 17.1.

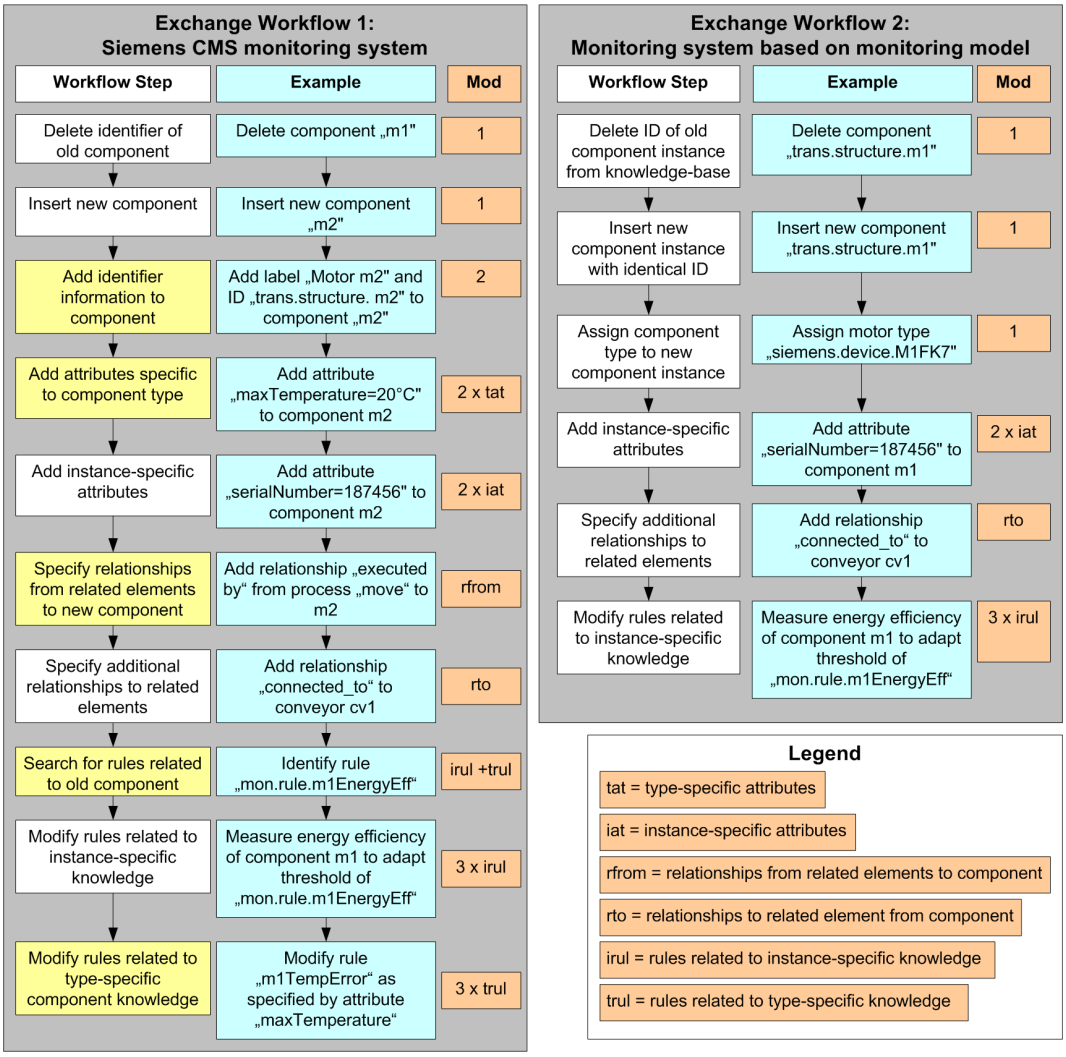


Figure 17.1.: Comparison of two exchange workflows in different settings

In a first step, the name of the component is deleted in workflow 2 and in a second

step a new component is specified. This is in contrast to exchange workflow 2 where the component's ID is deleted, but the new component is assigned with the same ID than the old component. ID's as defined in the concept model are unique within a specific plant by specifying a namespace of the plant, e.g. "trans.structure" to refer to the structure of a transportation plant. Thus, the component can be referenced explicitly by the same ID and the knowledge of the old component can be reused. Especially identifier information such as the name or label of the component can be reused in contrast to exchange workflow 1 where this information has to be specified in an additional step.

In the next two step, the component's attributes are defined. As shown in Section 8.1.3, the concept model distinguishes between instance-specific attributes *iat* and type-specific attributes *tat*. In usual monitoring systems, both kinds of attributes have to be adapted when a component is exchanged. But workflow 2 requires only the definition of *iat*, since type-specific attributes are inferred automatically by using the reference to the component template.

In a following step, relationships between different plant elements are adapted. The advantage of exchange workflow 2 is that relationships pointing to the ID of the old component *rfrom* persist. The step where these relationships need to be adapted as shown in workflow 1 can be skipped. Only the relationships pointing from the old component to related elements (*rto*) need to be added.

The last step is to adjust the monitoring rules to ensure a coherent monitoring of the new component. By means of the monitoring model, the relationship "refers-to" can be used to navigate from the monitored component to its rules. The advantage here is that (in a similar manner than attributes), the type-specific rules *trul* are automatically transferred to the new component. Only the instance-specific rules *irul* have to be modified or added.

Comparing the two settings with each other, the exchange workflow 1 requires some additional steps marked in yellow. These steps can be skipped when using the concept and monitoring model as a basis because they allow to infer the information needed within these additional steps automatically by means of axioms specified within the models.

The total amount of modifications for a step depends on the general changes *ch* to be conducted by the user and the number of objects *obj* that need to be modified or added, i.e. the number of attributes or relationships. To compute the total amount of modifications *Mod* the general changes have to be multiplied with the number of objects that need to be changed, such that $Mod = ch * obj$. For example, the changes needed to define a new instance-specific attribute "serial number" is 2, since first, the user has to select the attribute serial number and then he has to specify a value for the attribute. To compute the total amount of modifications Mod_{iat} needed to change or add three instance-specific attributes of a component is thus $Mod_{iat} = ch_{iat} * 3 = 6$. The total amount of modifications needed for exchange workflow 1 can thus be determined as:

$Mod_1 = 4 + 2 * iat + 2 * tat + rfrom + rto + 4 * irul + 4 * trul$

The total amount of modifications needed for exchange workflow 2 is:

$Mod_2 = 3 + 2 * iat + rto + 3 * irul$

The difference between both exchange workflows is then defined as:

$\Delta Mod = Mod_1 - Mod_2 = 1 + 2 * iat + 2 * tat + rfrom + irul + 4 * trul$

To compute the total difference, the amount of attributes, relationships referring from or to the component and rules of motor m1 for all AS implemented on the conveyor plant were determined. When exchanging the motor m1 most of the type-specific knowledge has to be modified, but the instance specific knowledge is rarely affected as the results presented in Table 17.1 show. The amount of modifications when using a standard condition monitoring system is $Mod_1 = 104$ and with the monitoring approach defined within this thesis $Mod_2 = 5$ The total difference between the two exchange workflow based on these results is thus $\delta Mod = 99$, which means that 99 additional modifications had to be made with a usual monitoring system compared to a monitoring system based on the monitoring model and the underlying conceptual modeling approach.

Application Scenario	*tat*	*rfrom*	*trul*	*iat*	*rto*	*irul*
AS1	1	1	9	1	0	0
AS2	2	2	2	0	0	0
AS3	0	1	2	0	0	0
AS4	4	1	4	0	0	0
AS5	1	1	2	0	0	0
Sum	8	6	19	1	0	0

Table 17.1.: Amount of tat, iat, rfrom, rto, irul and trul per AS

Part VI.

Finale

18. Evaluation of Research Questions

Based on a review of current literature regarding the initial problem specification, several research hypotheses and resulting research questions were formulated in Section 1.2.1. The research questions were evaluated throughout this thesis and the elaborated answers are summarized in this chapter.

Research Question 1: What factors influence whether knowledge-based resource monitoring systems in manufacturing companies can be established and how can these factors be addressed by means of an appropriate methodology?

Evaluation Question 1 (⇒ Chapter 5): Can the identified resource management factors crucial for the successful establishment of knowledge-based RMSs in the manufacturing domain be affirmed by similar case studies?

The RMS framework defined within this thesis arose from an empirical study carried out in a three-year collaboration with Siemens AG and Technische Universität München. By means of this study, important resource management factors and requirements for specifying a knowledge-based RMS in response to the demands of industrial practice were identified.

The findings were that multiple factors named “Resource Management Factors” need to be present for a successful establishment of a knowledge-based RMS in a manufacturing context. The identified Resource Management Factors could be confirmed entirely by similar case studies. Based on the identified Resource Management Factors, a knowledge engineering methodology was derived and requirements for RMS were specified to define an RMS in alignment with these constraints.

Research Question 2: How can knowledge-based techniques integrate knowledge on different plant aspects to realize a resource monitoring system, which is universally applicable independent of the branch of industry and the manufacturing engineering area?

Evaluation Question 2 (⇒ Chapter 9): Is the approach applicable to industrial plants of either distinct branches of industry or manufacturing engineering areas?

The analysis of two research projects RES-COM and I2MSteel in distinct manufacturing domains revealed that the conceptual modeling approach established as theoretical basis for implementing a knowledge-based RMS could be successfully applied on use cases in both domains. Further, partial plant models, such as the structure model and the process model, could be reused in both projects due to a degree of similarity of around 80%. By means of this evaluation, it was shown that the described approach can be applied to different kinds of industrial plants. This means that the proposed conceptual modeling approach is application-independent and can be used for all automation systems where knowledge of different engineering experts has to be integrated and analyzed, e.g. root cause analysis, condition monitoring, etc.

Research Question 3: What is the influence of an increase in different scalability factors (e.g. plant size or sampling rate) on the scalability and performance of a resource monitoring system realized by means of knowledge-based technologies?

Evaluation Question 3a (⇒ Chapter 12): Is the monitoring model proposed for the rule maintenance task scalable for reasonably large industrial plants?
Evaluation Question 3b (⇒ Chapter 16): To what degree is the computation time of the implemented resource monitoring system influenced by the sampling frequency or the amount of rules?

Performance tests to evaluate the scalability of the monitoring model for an increasing number of monitoring rules were performed to answer the evaluation question 3a. The response time for reasoning was used as the primary performance measure. The best response time could be accomplished by using the OWL DL reasoner HermiT for the reasoning task. The results proved that the proposed approach is scalable for reasonably large industrial plants.

The RMS was implemented and tested on a conveyor plant in various application scenarios to measure the computation time of the system when manipulating different scalability factors, such as the frequency rate and the number of rules. The computation time measured for the performance evaluation of evaluation question 3b was the time for computing a monitoring state based on monitoring input data. The test results showed a high system performance of the RMS with a computation time of less than 1 ms. An increase of the frequency rate or the number of rules did not influence the performance of the system. Such a high system performance could only be reached by means of an appropriate tool

chain, i.e. usage of the high performing CEP rule engine Drools for reasoning on plant data.

Research Question 4: What tools and procedures can be used to meet the RMS requirements and to improve ease-of-use through knowledge-based techniques to support plant engineering experts during the engineering process of a Resource Monitoring System?

Evaluation Question 4a (⇒ Chapter 13: Does the implemented plant model repository and state recognition module meet the RMS requirements?
Evaluation Question 4b (⇒ Chapter 14): Does the implemented RMS provide user guidance during engineering and maintenance tasks?

Wide-spread modeling tools and standards of semantic web, industrial and software engineering research communities were evaluated as alternatives for implementing the plant model repository (PMR) and the state recognition module (SRM) for providing an answer to evaluation question 4a. Throughout a precise examination of these tools and procedures, it was evaluated whether the RMS requirements specified in Section 7 could be fulfilled. The findings suggest that the requirements can only be entirely satisfied when using a semantic Wiki as basis for the PMR and CEP rule engines for the SRM. However, this investigation offers users the possibility to select an appropriate tool or procedure of the resulting list for implementing the RMS depending on user-specific requirements.

To answer evaluation question 4b, a small user study was conducted. The three usability criteria -operability, understandability and learnability- were rated by means of a questionnaire. An analysis of the questionnaire showed that users considered the proposed system easy to understand and learn. In addition, the users judged the operability of the software as very intuitive. Further, the efficiency of their modeling task, especially the time needed for implementation, could be improved by means of predefined modeling patterns and maintenance mechanisms. Nevertheless, a basic understanding of important Semantic Web concepts is required to understand the concepts introduced by the modeling framework of the RMS defined within this thesis entirely.

Research Question 5: How can knowledge-based techniques be used to improve reconfigurability of Resource Monitoring Systems?

Evaluation Question 5 (⇒ Chapter 17): What is the effort for modifications within the monitoring system if an installed monitored component in a plant is exchanged with a new component?

To evaluate the reconfigurability of the implemented RMS, the effort for modifications was considered. To this end, an exchange workflow containing all required modification steps required to exchange a plant component was defined. This workflow was once performed based on a usual condition monitoring system and once with the framework established within this thesis. The results showed that the total amount of modifications necessary is strongly dependent on type-specific information, e.g. type-specific component attributes. Applied on five application scenarios defined within the last chapter of this document, the amount of steps needed for reconfiguration of the system could be reduced by 99 steps when using the framework established within this thesis. Thus, the changeability of an RMS based on this framework can be effectively improved.

Summarizing this evaluation, the approach of this thesis answered the research questions and was thus able to address the desired research aims.

19. Conclusion

In this final chapter, the contributions and key findings of this thesis are summarized.

19.1. Core Contributions

The main objectives of this thesis have been addressed by means of contributions in different areas reflected by the structure of the parts II, III, IV and V. Within this structure, various aspects of research contributions have been addressed. The core objectives of this thesis reflect the main aspects of these contributions as summarized in the following.

19.1.1. Identification of Resource Management Factors

The empirical basis of the work presented in Chapter 4 identifies resource management factors that knowledge-based RMSs for industrial plants have to face. Based on this factors, a knowledge engineering methodology is derived that specifies how to gather and represent the knowledge needed for resource monitoring. This knowledge engineering methodology consists of four main tasks as shown in Chapter 6 and represents a systematic way to build a knowledge-based RMS.

Chapter 7 summarizes results of a requirements analysis. This analysis was conducted to create and categorize a comprehensive list of requirements necessary for addressing the Resource Management Factors and consequently for defining an RMS in alignment to demands of a modern manufacturing environment.

Placed in a broader context, the proposed knowledge engineering methodology and the requirements specified by means of the resource management factors can serve as a basis for establishing related knowledge-based resource management systems, such as diagnostics or control system, for manufacturing plants.

19.1.2. Integration of multiple plant models

Chapter 8 presented a conceptual modeling approach that allows for the integration of partial plant models defined by various domain experts of different engineering disciplines. The conceptual modeling approach is based on a MOF layering that allows for representation of plant aspects on four different modeling levels. The advantage of using such a metamodeling approach is on the one hand, a clear separation of the modeling levels to integrate all aspects of the industrial plant provided by the domain experts. On the other

hand, connections between the levels can be used to navigate between the levels, e.g. to navigate from a component instance “Motor m1” to its type, and to define constraints on usage and storage of the provided knowledge. These constraints can then be validated by appropriate reasoners. Furthermore, it has been shown that this approach is application-independent and can be used to describe industrial plants in different manufacturing domains

19.1.3. Engineering and maintenance of monitoring rules

A monitoring model for combining models and rule knowledge representation paradigms to organize a set of monitoring rules by structuring and reasoning about their constituents was proposed in Chapter 10. The monitoring model is instantiated on a semantic technology stack in Chapter 12. A plethora of technologies from the Semantic Web stack is used, such as XML-based format conversion for deploying RIF conform rules to runtime engines, RDF as a data exchange layer for interoperable representation of monitoring rule knowledge, OWL and SPARQL for validating and querying monitoring rule bases. This investigation has shown how to beneficially combine many otherwise rather disparate Semantic Web mechanisms in an integrated model for engineering and maintaining monitoring rules for a knowledge-based RMS.

19.1.4. Collaboration tool for deployment of a knowledge-based RMS

Another contribution of this work is to suggest a semantic collaboration tool with an appropriate infrastructure to support domain experts in their modeling tasks by allowing an advanced visualization and navigation of plant modeling entities needed for resource monitoring. Towards this end, Chapter 12 has demonstrated how a Semantic Mediawiki can be used as collaboration tool to address requirements of different engineering disciplines and user roles. This approach has been successfully applied to various use cases, which demonstrate the advantages of applying meta modeling techniques and semantic technologies for modeling in the manufacturing domain. By offering this conceptual modeling approach with its underlying infrastructure, the interdisciplinary cooperation of plant engineers during deployment of a knowledge-based RMS can be considerably improved as shown in a usability evaluation.

19.1.5. Resource Monitoring Application Scenarios

A prototypical implementation of the RMS on real industrial plants was used to illustrate different application scenarios for resource monitoring in Chapter 15. The ASs demonstrated the application of the resource monitoring framework using resource usage and process parameter profiles from machining experiments at various level on the functional plant hierarchy. All resource monitoring aspects considered throughout the entire thesis

were summarized and discussed within these ASs. These AS are an important contribution towards a more efficient usage of resources by machines and manufacturing systems. Finally, it could be shown that the deployed RMS is reconfigurable and scalable for real-life examples.

19.2. Outlook

This research work is one of the first investigations explicitly focusing on monitoring of resource by means of knowledge-based technologies. The importance of introducing resource monitoring systems in manufacturing is only gradually being increasingly considered by academia and industry. Thus, this research is an important step to pave the way for implementing resource monitoring systems in the manufacturing domain and has raised many questions in need of further investigation. Therefore, this section provides suggestions how the approach of this thesis might be enriched in the future.

Integration of data-driven approaches An integration of data-driven approaches into our framework would enable an increase in the system performance and the accuracy of the monitoring results in different ways. For example, methods of dimensionality reduction can be used to perform data analysis in production plants with high-dimensional sensor data sets. They provide a way to understand and visualize the structure of complex data sets. These methods also help to avoid phenomena like the curse of dimensionality or the empty space phenomenon [76]. Another advantage is that these techniques condense data so that the data can be processed more efficiently by the RMS. Robust principal component analysis could be used to identify outliers in the data, e.g. sensor failure or wrong processes.

Additionally, statistical classification methods, such as Bayesian networks, decision trees or support vector machines could be used. Statistical classification allows for identification of the categories to which new observations belong. By means of classification tools, e.g. Weka or RapidMiner, it is possible to learn categories for measured observations on the basis of a training set of data. This enables improvement of the monitoring rules over time by adapting the monitoring thresholds.

Automatic extraction of type-specific attributes A limitation in the research reported in this thesis is the current manual mapping and insertion of type-specific attributes of components. The type-specific attributes, such as *max rotational speed and max torque* of a motor, are specified in the manufacturer's product catalogs. Some manufacturers, e.g. Siemens, offer tools for extracting CAD data of component data specifications to store it in a CAD file [127]. These files could be used to automatically integrate type-specific attributes of the components in the RMS by mapping them on standard attributes in the predefined monitoring knowledge-base. Another advantage is that such an automatic data transfer to the RMS software minimizes potential sources of error.

Other research investigations in this area can further be used to extract this information out of documents directly. In [80], a method of constructing domain ontology using terminology processing and document retrieval is described. This approach could be used to enhance our system by automatically retrieve plant engineering data of various data sources.

Deployment of resource monitoring framework on projects of larger scale Currently, a first prototypical implementation of the resource monitoring system is deployed in small industrial plants for realizing different application scenarios. However, an important step to introduce the system in the manufacturing domain, is to test the system in industrial projects of larger scale. Firstly, the system performance and scalability could be verified and improved in a larger context. Secondly, this allows to gather more experiences about usability and acceptance of the overall system.

Integration of further ontologies To enhance the ontology framework provided as basis for the knowledge-based RMS, further models of related research applications could be integrated. For instance, to specify complex mathematical equations, the state recognition model defined in Section 10.2.2 could be extended by a mathematical model as described in [91]. Moreover, an energy ontology *OntoENERGY* defined to evaluate manufacturing systems by considering different energetic aspects is presented in [31]. By importing the energy ontology in the monitoring model, the energy efficiency application scenarios could be broadened to analyze energetic aspects within the manufacturing process steps. A diagnosis ontology proposed in [77], could further be used for identifying causes of excessive resource usage of plant elements detected by the resource monitoring system. The only effort is to adapt the ontologies prior to their integration into the knowledge-based of the RMS so that they stick to the concepts defined in the concept model.

19.3. Summary

The work reported in this thesis was guided by the overall objective of deploying a decentralized resource monitoring system by means of knowledge-based technologies in order to analyze the resource usage of plant elements. Providing an approach to monitor the resource usage of industrial plants, this thesis is an important step to improve resource efficiency in industrial automation. The approach presented in this thesis establishes sound insights into the resource usage behavior of industrial plants in the manufacturing domain. Moreover, the approach comprises a proposition for the technical implementation of the resource monitoring based on theoretical knowledge-based models so that resource savings potentials of manufacturing plants can actually be raised.

A. Questionnaire for Usability Evaluation

A.1. User Questionnaire

A.1.1. Semantic Framework

The semantic framework used throughout this user study is based on a Concept Model which is instantiated on a semantic technology stack. This semantic technology stack consists of a Semantic Mediawiki with underlying reasoning and querying tools.

A.1.2. User Experience

- How would you rate your previous experience in ontology engineering with Semantic Web technologies?

beginner	moderate	expert
☐	☐	☐

- How would you rate your previous experience in object oriented modeling with UML?

beginner	moderate	expert
☐	☐	☐

- How would you rate your previous experience in modeling of industrial plants with CAEX?

beginner	moderate	expert
☐	☐	☐

- If applicable, which modeling approaches did you use already prior to this experiment?
  - Progété: ☐
  - SysML: ☐
  - CAEX: ☐

A.1.3. Studies for Users on Level M2

Tasks

- **Task 1:** The participants were asked to add some additional modeling entities of their meta models (e.g. service model, product model, context model) by using predefined templates for "Entity Metatypes" as specified by our concept model.
- **Task 2:** The participants were asked to define semantic relationship between their modeling entities by using predefined templates for "Relationship Metatypes" as specified by our concept model.

Operability

- How were the difficulties you experienced to complete task 1?

low	average	high
☐	☐	☐

- How were the difficulties you experienced to complete task 2?

low	average	high
☐	☐	☐

- How was the tool support provided by predefined modeling templates & forms, consistency checks, namespaces and browsing ability during the two tasks?

very inadequate	inadequate	good	very good
☐	☐	☐	☐

- Did the consistency checks and queries support you in executing different modeling tasks?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- What was the main obstacle that you found during the tasks?

Learnability

- How much time (working hours) did you take approximately for getting acquainted with the conceptual modeling approach? .....
- How much time (working hours) did you take approximately for learning how to use the modeling concepts of the concept model, e.g. Entity Type, Relationship Type, etc.? .....

Understandability

- Were the different explications and examples helpful to understand the concepts of the conceptual model?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- Did you find the navigation between different levels of abstractions intuitive, e.g. to navigate from an Entity Type to an Entity Instance or an Attribute to an Attribute Value?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- Did you find the navigation between different plant aspects intuitive, e.g. navigation from structural to process model?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- Are you considering the proposed conceptual modeling approach usefull for modeling different aspects (such as the structure, the processes, etc.) of industrial plants?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- Are the required actions for creating modeling entities, e.g. a Hardware Component Instance or a Process Instance, clear and intuitive?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

A.1.4. Studies for Users on Level M0 & M1

Tasks

- **Task 1:** The participants were asked to define a new component type, e.g. a motor type, of a specific plant component via the form "Entity Type".
- **Task 2:** The participants were asked to define a new component instance with the form "Entity Instance" and connect it to another component instance.
- **Task 3:** The participants were asked to exchange an existing component instance. This component instance had a connection to its component type and to another component which had to be modified.

A.1.5. Operability

- How were the difficulties you experienced to complete task 1?

low	average	high
☐	☐	☐

- How were the difficulties you experienced to complete task 2?

low	average	high
☐	☐	☐

- How were the difficulties you experienced to complete task 3?

low	average	high
☐	☐	☐

- How was the tool support provided by predefined modeling templates & forms, consistency checks, namespaces and browsing ability during the two tasks?

very inadequate	inadequate	good	very good
☐	☐	☐	☐

- Did the consistency checks and queries support you in executing different modeling tasks?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- What was the main obstacle that you found during the tasks?

Learnability

- How much time (working hours) did you take approximately for getting acquainted with the conceptual modeling approach?
- How much time (working hours) did you take approximately for learning how to use the modeling concepts of the concept model, e.g. Entity Type, Relationship Type, etc.?

Understandability

- Did you find the navigation between different levels of abstractions intuitive, e.g. from a motor instance to its motor type?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- Did you find the navigation between different plant aspects intuitive, e.g. navigation to related components?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- Are you considering the proposed tool and modeling approach usefull for modeling different aspects of industrial plants, e.g. to model the connections between plant components and to relate them with knowledge of the component types?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

- Are the required actions for creating modeling entities, e.g. a motor instance or an instance of a process, clear and intuitive?

not at all	not really	rather yes	absolutely
☐	☐	☐	☐

A.2. Outcomes of Questionnaire

A.2.1. User Experience

- How would you rate your previous experience in ontology engineering with Semantic Web technologies?

beginner	moderate	expert
2	1	1

- How would you rate your previous experience in object oriented modeling with UML?

beginner	moderate	expert
0	4	0

- How would you rate your previous experience in modeling of industrial plants with CAEX?

beginner	moderate	expert
2	1	1

- If applicable, which modeling approaches did you use already prior to this experiment?
  - Progété: 1
  - SysML: 3
  - CAEX: 2

A.2.2. Studies for Users on Level M2

Operability

- How were the difficulties you experienced to complete task 1?

low	average	high
3	1	0

- How were the difficulties you experienced to complete task 2?

low	average	high
3	1	0

- How was the tool support provided by predefined modeling templates & forms, consistency checks, namespaces and browsing ability during the two tasks?

very inadequate	inadequate	good	very good
0	1	3	0

- Did the consistency checks and queries support you in executing different modeling tasks?

not at all	not really	rather yes	absolutely
0	0	3	1

- What was the main obstacle that you found during the tasks?
  - Not yet visualization
  - Learning curve
  - get used to the complexity of the tasks

Learnability

- How much time (working hours) did you take approximately for getting acquainted with the conceptual modeling approach? 3/10/8/6
- How much time (working hours) did you take approximately for learning how to use the modeling concepts of the concept model, e.g. Entity Type, Relationship Type, etc.? 5/20/8/7

Understandability

- Were the different explications and examples helpful to understand the concepts of the conceptual model?

not at all	not really	rather yes	absolutely
0	0	2	2

- Did you find the navigation between different levels of abstractions intuitive, e.g. to navigate from an Entity Type to an Entity Instance or an Attribute to an Attribute Value?

not at all	not really	rather yes	absolutely
0	1	1	2

- Did you find the navigation between different plant aspects intuitive, e.g. navigation from structural to process model?

not at all	not really	rather yes	absolutely
0	0	3	1

- Are you considering the proposed conceptual modeling approach usefull for modeling different aspects (such as the structure, the processes, etc.) of industrial plants?

not at all	not really	rather yes	absolutely
0	0	1	3

- Are the required actions for creating modeling entities, e.g. a Hardware Component Instance or a Process Instance, clear and intuitive?

not at all	not really	rather yes	absolutely
0	1	2	1

A.2.3. Studies for Users on Level M0 & M1

Operability

- How were the difficulties you experienced to complete task 1?

low	average	high
3	1	0

- How were the difficulties you experienced to complete task 2?

low	average	high
2	2	0

- How were the difficulties you experienced to complete task 3?

low	average	high
1	3	0

- How was the tool support provided by predefined modeling templates & forms, consistency checks, namespaces and browsing ability during the two tasks?

very inadequate	inadequate	good	very good
0	0	4	0

- Did the consistency checks and queries support you in executing different modeling tasks?

not at all	not really	rather yes	absolutely
0	1	2	1

- What was the main obstacle that you found during the tasks?
  - not yet a visualization
  - Learning curve
  - attribute heritage
  - modeling of rules within the Wiki

Learnability

- How much time (working hours) did you take approximately for getting acquainted with the conceptual modeling approach? 2/10/8/6
- How much time (working hours) did you take approximately for learning how to use the modeling concepts of the concept model, e.g. Entity Type, Relationship Type, etc.? 5/20/8/7

Understandability

- Did you find the navigation between different levels of abstractions intuitive, e.g. from a motor instance to its motor type?

not at all	not really	rather yes	absolutely
0	2	1	1

- Did you find the navigation between different plant aspects intuitive, e.g. navigation to related components?

not at all	not really	rather yes	absolutely
0	1	2	1

- Are you considering the proposed tool and modeling approach usefull for modeling different aspects of industrial plants, e.g. to model the connections between plant components and to relate them with knowledge of the component types?

not at all	not really	rather yes	absolutely
0	0	3	1

- Are the required actions for creating modeling entities, e.g. a motor instance or an instance of a process, clear and intuitive?

not at all	not really	rather yes	absolutely
0	0	3	1

Bibliography

1. L. Abele, M. Anic, T. Gutmann, J. Folmer, M. Kleinsteuber, and B. Vogel-Heuser. Combining Knowledge Modeling and Machine Learning for Alarm Root Cause Analysis. In *IFAC Conference on Manufacturing, Modeling and Control (MIM13)*, number 2007. 2013.

2. L. Abele and S. Grimm. Knowledge-based Integration of Industrial Plant Models. In *Proceedings of Conference of the IEEE Industrial Electronics Society (IECON)*. 2013.

3. L. Abele, T.r. Hansen, and M. Kleinsteuber. A knowledge engineering methodology for resource monitoring in the industrial domain. In *IFAC Conference on Manufacturing, Modeling and Control (MIM13)*, pp. 0–5, 2013.

4. L. Abele, M. Kleinsteuber, and T. Hansen. Resource Monitoring in Industrial Production with Knowledge-Based Models and Rules. In *Proceedings of PhD Workshop on ACM Conference on Information and Knowledge Management (CIKM)*. 2011.

5. L. Abele, C. Legat, S. Grimm, and A.W. Müller. Ontology-based Validation of Plant Models. In *IEEE International Conference on Industrial Informatics*. 2013.

6. L. Abele, L. Ollinger, I. Heck, and M. Kleinsteuber. A Decentralized Resource Monitoring System Using Structural , Context and Process Information. In *Trends in Inteligent Robotics, Automation and Manufacturing (IRAM)*. Springer Berlin Heidelberg, 2012.

7. I. Alexander and N. Maiden. Scenarios, stories, and use cases: the modern basis for system development. In *Computing & Control Engineering Journal*, 15(5), pp. 24–29, 2004.

8. D. Anicic and P. Fodor. EP-SPARQL : A Unified Language for Event Processing and Stream Reasoning. In *Proceedings of the 20th International Conference on World Wide Web*, pp. 635–644. 2011.

9. AutomationMLConsortium. *AutomationML Whitepaper Part 1 - Architecture and general requirements*. Technical Report, 2013. URL `https://www.automationml.org`.

10. N. Avenue and R. May. Automated functional design of engineering systems. In *Journal of Intelligent Manufacturing*, 13(2), pp. 119–133, 2002.

11. L. Badea. Reifying Concepts in Description Logics. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 142–147. 1997.

12. M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. In *Information Systems*, 2(4), 2007.

13. D. Batra and N. Wishart. Comparing a rule-based approach with a pattern-based approach at different levels of complexity of conceptual data modelling tasks. In *International Journal of Human-Computer Studies*, 61(4), pp. 397–419, 2004.

14. J. Bernard. Use of a rule-based system for process control. In *IEEE Control Systems Magazine*, 8(5), pp. 3–13, 1988.

15. BMBF. RES-COM - Resource Conservation through Context-dependent Machine-to-Machine Communication. http://www.res-com-project.org, 2011.

16. J. Bock, P. Haase, Q. Ji, and R. Volz. Benchmarking OWL Reasoners. In *Proceedings of the ARea2008 Workshop*. 2008.

17. W. Bohlken and B. Neumann. Generation of Rules from Ontologies for High-Level Scene Interpretation. In *Proceedings of the International Symposium on Rule Interchange and Applications*. 2009.

18. D. Brandi. Cut costs with manufacturing IT standards, best practices. In *Control Engineering*, 56(4), pp. 28–28, 2009.

19. BRD. *Nationale Nachhaltigkeitsstrategie - Fortschrittsbericht 2012*. Technical Report, Bundesregierung Deutschland, 2012.

20. C. Brecher, W. Herfs, D. Özdemir, and A. Müller. Integration und Durchgängigkeit von Information im Produktionsmittellebenszyklus. In *ZWF - Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 105, pp. 271–276, 2010.

21. J. Breuker. A cognitive science perspective on knowledge acquisition. In *International Journal of Human-Computer Studies*, 71(2), pp. 177–183, 2013.

22. C. Bunks, D. McCarthy, and A.A. TARIK. Condition-based maintenance of machines using Hidden Markov Models. In *Mechanical Systems and Signal Processing*, 14(4), pp. 597–612, 2000.

23. K. Bunse, M. Vodicka, P. Schönsleben, M. Brülhart, and F.O. Ernst. Integrating energy efficiency performance in production management – gap analysis between industrial needs and scientific literature. In *Journal of Cleaner Production*, 19(6-7), pp. 667–679, 2011.

24. M. Calder, R. Morris, and F. Peri. Machine reasoning about anomalous sensor data. In *Ecological Informatics*, 5(1), pp. 9–18, 2010.

25. P. Caramia, Massimiliano, Dell'Olmo. *Effective Resource Management in Manufacturing Systems - Optimization Algorithms for Production*. Springer London, 2006.

26. C. Casagni, C.D. Francescomarino, M. Dragoni, L. Fiorentini, L. Franci, M. Gerosa, C. Ghidini, F. Rizzoli, M. Rospocher, A. Rovella, L. Serafini, S. Sparaco, and A. Tabarroni. Wiki-based conceptual modeling: an experience with the Public Administration. In *International Semantic Web Conference (ISWC11)*, number 1, pp. 17–32. 2011.

27. E. Castillo, J. Gutiérrez, and A. Hadi. *Expert Systems and Probabilistic Network Models*. Springer New York, 1997.

28. O. Castillo and P. Melin. Fuzzy logic for plant monitoring and diagnostics. In *IEEE International Conference on Fuzzy Systems*, volume 1, pp. 61–66. 2004.

29. J. Chen and C.M. Liao. Dynamic process fault monitoring based on neural network and PCA. In *Journal of Process Control*, 12(2), pp. 277–289, 2002.

30. L. Chiang, E. Russell, and R. Braatz. *Fault Detection and Diagnosis in Industrial Systems*. Springer London, 2001.

31. L. Christiansen, T. Linnenberg, A. Fay, C. Seitz, and A. Müller. Energietechnische Beschreibung fertigungstechnischer Prozesse zur Bewertung der Energieeffizienz. In *Tagungsband "Automation 2013"*. 2013.

32. L. Christiansen, A. Fay, B. Opgenoorth, and J. Neidig. Improved Diagnosis by Combining Structural and Process Knowledge. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*. 2011.

33. L. Christiansen, T. Linnenberg, A. Fay, C. Seitz, and A.W. Müller. Energieeffizienz in der Fertigung bewerten - Ontologiebasierte Beschreibung und Simulation. In *atp edition*, pp. 70–77. 2013.

34. G. Chryssolouris. *Manufacturing Systems: Theory and Practice*. 2nd edition. Springer New York, 2006.

35. N. Chungoora and R. Young. A framework to support semantic interoperability in product design and manufacture. In *Global Product Development*, pp. 435–443, 2011.

36. W. Dai, V.N. Dubinin, and V. Vyatkin. Automatically Generated Layered Ontological Models for Semantic Analysis of Component-Based Control Systems. In *IEEE Transactions on Industrial Informatics*, 9(4), pp. 2124–2136, 2013.

37. A. Decelle. Towards a unified specification of the construction process information, the PSL approach. In *Proceedings of the European Conference of Product and Process Modelling (ECPPM)*. 2000.

38. T. Devoldere, W. Dewulf, W. Deprez, B. Willems, and J. Duflou. Improvement potential for energy consumption in discrete part production machines. In *Advances in Life Cycle Engineering for Sustainable Manufacturing Businesses*, pp. 311–316. Springer London, 2007.

39. A.K. Dey. Understanding and Using Context. In *Personal Ubiquitous Computing*, 5(1), pp. 4–7, 2001.

40. A. Dietmair and A. Verl. Energy consumption modeling and optimization for production machines. In *IEEE International Conference on Sustainable Energy Technologies, ICSET*, pp. 574–579. 2008.

41. R. Drath. *Datenaustausch in der Anlagenplanung mit AutomationML - Integration von CAEX, PLCopen XML und COLLADA*. Springer Berlin Heidelberg, 2010.

42. S. Ebersbach and Z. Peng. Expert system development for vibration analysis in machine condition monitoring. In *Expert Systems with Applications*, 34(1), pp. 291–299, 2008.

43. C.M. Eckert, M.K. Stacey, and P.J. Clarkson. The spiral of applied research: A methodological view on integrated design research. In *Proceedings of the 14th International Conference on Engineering Design (ICED'03)*. 2003.

44. H.A. ElMaraghy. *Changeable and Reconfigurable Manufacturing Systems*. Springer London, 2011.

45. E. Estévez and M. Marcos. Model-Based Validation of Industrial Control Systems. In *IEEE Transactions on Industrial Informatics*, 8(2), pp. 302–310, 2012.

46. EuropeanParliament. *Proceedings of the Workshop on "Resource Efficiency"*. Technical Report, European Parliament, Brussels, 2012.

47. D.P. Filev, R.B. Chinnam, F. Tseng, and P. Baruah. An Industrial Strength Novelty Detection Framework for Autonomous Equipment Monitoring and Diagnostics. In *IEEE Transactions on Industrial Informatics*, 6(4), pp. 767–779, 2010.

48. C.L. Forgy. Rete : A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In *Artifical Intelligence*, 19, pp. 17–37, 1982.

49. Frank Manola and E. Miller. RDF Primer. 2004. URL `http://www.w3.org/TR/rdf-primer/`.

50. A.A. Garza, J.J. Serrano, R.O. Carot, K. Ramirez, J.M. García, H. Arias, and J. Soria. Monitoring and Diagnostics with Intelligent Agents Using Fuzzy Logic. In *Engineering Letters*, 15(1), pp. 105–111, 2007.

51. GCT. Innovationsallianz Green Carbody Technologies. 2010. URL `http://www.greencarbody.de/`.

52. O. Geramifard, J.x. Xu, J.h. Zhou, and X. Li. A Physically Segmented Hidden Markov Model Approach for Continuous Tool Condition Monitoring: Diagnostics and Prognostics. In *IEEE Transactions on Industrial Informatics*, 8(4), pp. 964–973, 2012.

53. B. Glimm and P. Bijan. *SPARQL 1.1 Entailment Regimes*. Technical Report, W3C, 2010. URL `http://www.w3.org/TR/2010/WD-sparql11-entailment-20100126/`.

54. H.T. Grimmelius, P.P. Meiler, H. Maas, B. Bonnier, J.S. Grevink, and R.F. van Kuilenburg. Three state-of-the-art methods for condition monitoring. In *IEEE Transactions on Industrial Electronics*, 46(2), pp. 407–416, 1999.

55. Y. Grobshtein, V. Perelman, E. Safra, and D. Dori. Systems Modeling Languages: OPM Versus SysML. In *Proceedings on IEEE International Conference on Systems Engineering and Modeling*, pp. 102–109. 2007.

56. T.R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *International Journal of Human-Computer Studies*, pp. 907–928. Kluwer Academic Publishers, 1993.

57. M. Gruninger and C. Menzel. The Process Specification Language (PSL) - Theory and Applications. In *Artifical Intelligence Magazine*, 24(3), pp. 63–73, 2003.

58. K. Güttel and A. Fay. Beschreibung von fertigungstechnischen Anlagen mittels CAEX. In *Automatisierungstechnische Praxis*, Heft 5, pp. 34–39, 2008.

59. R. Hiersemann. Integration von Energiemonitoring-Funktionalitäten in Anlagenleitständen - Möglichkeiten und Grenzen. In *Karlsruher Leittechnisches Kolloquium*, pp. 139–151, 2010.

60. F. van Houten, T. Tomiyama, and O.W. Salomons. Product Modelling for Model-Based Maintenance. In *CIRP Annals - Manufacturing Technology*, 47(1), pp. 123–128, 1998.

61. S. Hu, F. Liu, Y. He, and T. Hu. An on-line approach for energy efficiency monitoring of machine tools. In *Journal of Cleaner Production*, 27, pp. 133–140, 2012.

62. I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press, 1992.

63. S.R. Jakuba. Failure mode and effect analysis. In *Maximizing Machinery Uptime*, volume 5 of *Practical Machinery Management for Process Plants*, pp. 135–155. Gulf Professional Publishing, 2006.

64. A. Jameson. Usability and the Semantic Web. In *Proceedings of the European Semantic Web Conference (ESWC)*, Lecture Notes in Computer Science, pp. 3–11. 2006.

65. A.K. Jardine, D. Lin, and D. Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. In *Mechanical Systems and Signal Processing*, 20(7), pp. 1483–1510, 2006.

66. JbossCummunity. Drools Documentation - Chapter 5. 2013. URL `http://docs.jboss.org/drools/release/5.4.0.Final/drools-planner-docs/html/scoreCalculation.html`.

67. X. Jiang. *High-Fidelity Power Metering and Run-Time Energy Management in Wireless Sensor Networks*. Ph.D. thesis, B.S. University of California, Berkeley, 2004.

68. E.S.S.G..C. KG. EPLAN - efficient engineering. 2013. URL `http://www.eplan.de/en/start/`.

69. G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. 2004. URL `http://www.w3.org/TR/rdf-concepts/`.

70. A. Kolk and R. van Tulder. International business, corporate social responsibility and sustainable development. In *International Business Review*, 19(1), pp. 119–125, 2010.

71. M. Kroetzsch, D. Vrandecic, and M. Voelkel. Semantic MediaWiki. In *Proceedings of 5th International Semantic Web Conference*, pp. 935–942. 2006.

72. P. L, N. Ivezic, and C. Schlenoff. Ontology engineering for distributed collaboration in manufacturing. In *Proceedings of the AIS2000 conference*. 2000.

73. I. Lake, G.J.D.G. Gil, D.B. Alabi, O.E. Iyun, and N.F. Thornhill. Merging Process Models and Plant Topology. In *International Symposium on Advanced Control of Industrial Processes (ADCONIP)*, pp. 15–21. 2011.

74. G. Laleci, G. Aluc, A. Dogac, A. Sinaci, O. Kilic, and F. Tuncer. A semantic backend for content management systems. In *Knowledge-Based Systems*, 23(8), pp. 832–843, 2010.

75. J.H. Lee, S.J. Fenves, C. Bock, H.w. Suh, S. Rachuri, X. Fiorentini, and R.D. Sriram. A Semantic Product Modeling Framework and Its Application to Behavior Evaluation. In *IEEE Transaction on Automation Science and Engineering*, 9(1), pp. 110–123, 2012.

76. J.A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer New York, 2007.

77. C. Legat, T.M. Hubauer, and C. Seitz. Integrated Diagnosis for Adaptive Service-oriented Manufacturing Control with Autonomous Products. In *International Conference on Industrial Engineering and Systems Management*. 2011.

78. C. Legat, S. Lamparter, and B. Vogel-Heuser. Knowledge-based Technologies for Future Factory Engineering and Control. In *Service Orientation in Holonic and Multi-agent Manufacturing and Robotics*, volume 472, chapter 23, pp. 355–374. Springer Berlin Heidelberg, 2013.

79. C. Legat, J. Neidig, and M. Roshchin. Model-based Knowledge Extraction for Automated Monitoring and Control. In *IFAC World Congress*, pp. 5225–5230. IFAC, 2011.

80. S.y. Lim, M.h. Song, K.j. Son, and S.j. Lee. Domain ontology construction based on semantic relation information of terminology. In *Proceedings of Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 2213–2217. 2004.

81. H. Lin and J. Harding. A manufacturing system engineering ontology model on the semantic web for inter-enterprise collaboration. In *Computers in Industry*, 58, pp. 428–437, 2007.

82. M. Loskyll. *Entwicklung einer Methodik zur dynamischen kontextbasierten Orchestrierung semantischer Feldgeraetefunktionalitaeten*. Phd-thesis, TU Kaiserslautern, 2013.

83. M. Loskyll, J. Schlick, S. Hodek, L. Ollinger, T. Gerber, B. Pirvu, and T.G. De. Semantic Service Discovery and Orchestration for Manufacturing Processes. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–8. 2011.

84. M. Botts and A. Robin. *Sensor Model Language (SensorML) Implementation Specification 1.0.0*. Technical Report, 2007.

85. P. Mariño, S. Member, F. Poza, S. Otero, and M.A. Domínguez. Reconfigurable Industrial Sensors for Remote Condition Monitoring and Modeling. In *IEEE Transactions on Industrial Electronics*, 57(12), pp. 4199–4208, 2010.

86. J. Martin and J.J. Odell. *Object-Oriented Methods: A Foundation, UML Edition*. 2nd edition. Prentice Hall PTR, 1994.

87. S. Mechs. *Model-based Engineering for Energy-Efficient Operation of Factory Automation Systems within Unproductive Phases*. Ph.D. thesis, Technische Universitaet Clausthal, 2013.

88. S. Mechs, S. Grimm, D. Beyer, and S. Lamparter. Evaluation of prediction accuracy for energy-efficient switching of automation facilities. In *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 6928–6933. 2013.

89. M. Milanović, D. Gašević, and A. Giurca. Model Transformations to Bridge Concrete and Abstract Syntax of Web Rule Languages. In *Computer Science and Information Systems*, 6(2), pp. 47–58, 2009.

90. R. Milne. TIGER: Knowledge Based Gas Turbine Condition Monitoring. In *IEE Colloquium on Artificial Intelligence-Based Applications for the Power Industry*. 1999.

91. J. Morbach, A. Yang, and W. Marquardt. *Mathematical Models*. Technical Report, 2008.

92. T. Moser, W.D. Sunindyo, and S. Biffl. Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas. In *International Conference on Software Engineering & Knowledge Engineering (SEKE2009)*, pp. 233–239. 2009.

93. V.S.K. Murthy Balijepalli, V. Pradhan, S.A. Khaparde, and R.M. Shereef. Review of demand response under smart grid paradigm. In *Proceedings of IEEE Conference on Innovative Smart Grid Technologies (ISGT)*, pp. 236–243. 2011.

94. Nachhaltigkeitsstrategie Bundesregierung Deutschland. *Nationale Nachhaltigkeitsstrategie Fortschrittsbericht 2012*. Technical Report, 2012.

95. A. Naumenko, S. Nikitin, V. Terziyan, and A. Zharko. Strategic Industrial Alliances in Paper Industry : XML- vs . Ontology-Based Integration. In *The Learning Organization*, 12(5), pp. 492 – 514, 2005.

96. R.. Neukirchner and F. Fuchs. MES als Werkzeug in einem LeanSigma-Projekt zur Steigerung der Energieeffizienz. In *Karlsruher Leittechnisches Kolloquium 2010*, pp. 153–163, 2010.

97. N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Fergerson, and M. Musen. Creating Semantic Web contents with Protege-2000. In *IEEE Intelligent Systems*, 16(2), pp. 60–71, 2000.

98. Object Management Group. In MOF (Meta Object Facility) Core Specification 1.4, (04), 2004.

99. L. Ollinger, J. Schlick, and S. Hodek. Konzeption und praktische Anwendung serviceorientierter Architekturen in der Automatisierungstechnik. In *VDI Automatisierungskongress (AUTOMATION-2011)*. 2011.

100. OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C, 2012. URL `http://www.w3.org/TR/owl2-overview/`.

101. A. Pakonen, T. Tommila, and J. Hirvonen. A fuzzy ontology based approach for mobilising industrial plant knowledge. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–8. 2010.

102. A. Pakonen, T. Tommila, T. Pirttioja, and I. Seilonen. OWL based information agent services for process monitoring. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 9–16. 2007.

103. P. Palensky and D. Dietrich. Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads. In *IEEE Transactions on Industrial Informatics*, 7(3), pp. 381–388, 2011.

104. S. Park and J. Kang. Using Rule Ontology in Repeated Rule Acquisition from Similar Web Sites. In *IEEE Transactions on Knowledge and Data Engineering*, 24(6), pp. 1106–1119, 2012.

105. P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. 2004. URL `http://www.w3.org/TR/owl-semantics/`.

106. M. Proctor. Drools: A Rule Engine for Complex Event Processing. In *Applications of Graph Transformations with Industrial Relevance*, volume 7233. Springer Berlin Heidelberg, 2012.

107. Production Systems Technologies, Inc. Benchmarking CLIPS/R2. URL `http://www.pst.com/benchcr2.htm`.

108. R. Raskin. Semantic Web for Earth and Environmental Terminology (SWEET). 2011. URL `http://sweet.jpl.nasa.gov/ontology/`.

109. M. Ritthoff, C. Liedtke, and T. Merten. *Der Werkstoff Stahl und seine Anwendung - Verbesserung von Rohstoffproduktivität und Ressourcenschonung*. Technical Report, Wuppertal Institut für Klima, Umwelt, Energie GmbH, 2007.

110. S. Runde and A. Fay. Software Support for Building Automation Requirements Engineering - An Application of Semantic Web Technologies in Automation. In *IEEE Transactions on Industrial Informatics*, 7(4), pp. 723–730, 2011.

111. S. Runde, A. Fay, S. Schmitz, and U. Epple. Wissensbasierte Systeme im Engineering der Automatisierungstechnik. In *at - Automatisierungstechnik*, 59(1), pp. 42–49, 2011.

112. S. Runde, K. Güttel, and A. Fay. Transformation von CAEX-Anlagenplanungsdaten in OWL- Eine Anwendung von Technologien des Semantic Web. In *Tagungsband Automation*, 2009.

113. M. Sarcar, K. Mallikarjuna, and K.L. Narayan. *Computer Aided Design and Manufacturing*, volume 93. Prentice-Hall of India Private Limited, New Delhi, 2008.

114. M. Schleburg, L. Christiansen, A. Fay, and N.F. Thornhill. Reduction of alerts in automated systems based on a combined analysis of process connectivity and alarm

logs. In *Tagungsband der "European Symposium on Computer Aided Process Engineering" (ESCAPE)*, 2012.

115. M. Schleipen and M. Okon. The CAEX tool suite - User assistance for the use of standardized plant engineering data exchange. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–7. 2010.

116. M. Schleipen, R. Drath, and O. Sauer. The system-independent data exchange format CAEX for supporting an automatic configuration of a production monitoring and control system. In *IEEE International Symposium on Industrial Electronics*, pp. 1786–1791. 2008.

117. C. Schlenoff, M. Gruninger, F. Tissot, and J. Lee. *The Process Specification Language (PSL) Overview and Version 1.0 Specification*. Technical Report, 2000.

118. T. Schmidberger and A. Fay. A rule format for industrial plant information reasoning. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 360–367. 2007.

119. S. Schmitz, M. Schluetter, and U. Epple. Automation of Automation – Definition , Components and Challenges. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2009.

120. M. Schneider. Towards a General Object Memory. In *Artificial Intelligence*, 2007.

121. C. Seitz, C. Legat, Z. Liu, S. Ag, and C. Technology. Flexible Manufacturing Control with Autonomous Product Memories. In *Proceedings of IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2010.

122. C. Seitz and T. Schöler. An Agent-based Sensor Middleware for generating and interpreting Digital Product Memories. In *Communication*, 2009.

123. Siemens. Comos at a glance - plant engineering software. .

124. Siemens. SIMOTICS S-1FK7 Servomotors. . URL `https://c4b.gss.siemens.com/resources/images/articles/6zb5711-0at02-0aa3.pdf`.

125. Siemens. *Safety Integrity Level (SIL) according to EN 62061*, 2010.

126. Siemens. PLM - Product Lifecycle Management : Siemens PLM Software - Germany. 2013. URL `http://www.plm.automation.siemens.com/de_de/`.

127. Siemens. SIMATIC CAx-Data - Automation Software - Siemens. 2013. URL `http://www.automation.siemens.com/mcms/automation-software/en/shared-tasks/documentation/cax-data/pages/default.aspx`.

128. E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. In *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2), pp. 51–53, 2007.

129. R. Sivakumar and P.V. Arivoli. Ontology Visualization Protege Tools - A Review. In *International Journal of Advanced Information Technology*, 1(4), 2011.

130. L. Sivill, J. Manninen, and P. Ahtila. A combined approach to energy efficiency monitoring in the pulp and paper industry. In *Proceedings of the 22nd International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems*, pp. 1513–1521. 2009.

131. L. Sivill, J. Manninen, I. Hippinen, and P. Ahtila. Success factors of energy management in energy- intensive industries : Development priority of energy performance measurement. In *International Journal of Energy Research*, 37(8), pp. 936–951, 2012.

132. D. Solomatine, L. See, and R. Abrahart. *Data-Driven Modelling: Concepts, Approaches and Experiences*, volume 68 of *Water Science and Technology Library*. Springer Berlin Heidelberg, 2008.

133. A. Sreenath and S. Venkatesh. Experimental studies on the wear of engine components. In *Wear*, 16(4), pp. 245–254, 1970.

134. C. Stammen. *Condition-Monitoring für intelligente hydraulische Linearantriebe*. Ph.D. thesis, RWTH Aachen, 2005.

135. T. Strasser, C. Sünder, and A. Valentini. Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector. In *Proceedings of International Conference on Industrial Informatics (INDIN)*. 2008.

136. G. Strbac. Demand side management: Benefits and challenges. In *Energy Policy*, 36(12), pp. 4419–4426, 2008.

137. H. Su and K.T. Chong. Induction Machine Condition Monitoring Using Neural Network Modeling. In *IEEE Transactions on Industrial Electronics*, 54(1), pp. 241–249, 2007.

138. A. Theorin, L. Ollinger, and C. Johnsson. Service-oriented Process Control with Grafchart and the Devices Profile for Web Services. In *Proceedings of the IFAC Symposium on Information Control Problems in Maufacturing (INCOM))*. 2012.

139. K. Thramboulidis. Model-Integrated Mechatronics -Toward a New Paradigm in the Development of Manufacturing Systems. In *IEEE Transactions on Industrial Informatics*, 1(1), pp. 54–61, 2005.

140. K. Thramboulidis, D. Perdikis, and S. Kantas. Model driven development of distributed control applications. In *International Journal of Advanced Manufacturing Technologies*, 33(3-4), pp. 233–242, 2006.

141. K. Thramboulidis. The 3+1 SysML View-Model in Model Integrated Mechatronics. In *Journal of Software Engineering & Applications*, 03(02), pp. 109–118, 2010.

142. Till Schmidberger. *Wissensbasierte Auswertung von Anlagenplanungsdaten für die Unterstützung des Prozessleittechnikingenieurs*. Ph.D. thesis, Helmut Schmidt Universität Hamburg, 2008.

143. M. Tokhi, M. Hossain, M. Baxter, and P. Fleming. Performance Evaluation Issues in Real-Time Parallel Signal Processing and Control. In *Journal of Parallel and Distributed Computing*, 42(1), pp. 67–74, 1997.

144. P. Tzscheutschler. Interview with Dr.- Ing. Peter Tzscheutschler, Technische Universität München. 2011.

145. V. Uraikul, C.W. Chan, and P. Tontiwachwuthikul. Artificial intelligence for monitoring and supervisory control of process systems. In *Engineering Application of Artificial Intelligence*, 20(2), pp. 115–131, 2007.

146. J. Verner, J. Sampson, V. Tosic, N.A. Bakar, and B. Kitchenham. Guidelines for industrially-based multiple case studies in software engineering. In *Proceeding of International Conference on Research Challenges in Information Science*, pp. 313–324, 2009.

147. M. Viinikkala, S. Syrjälä, and S. Kuikka. A Maintenance Demand Analyzer – a Web Service based on a Semantic Plant Model. In *IEEE International Conference on Industrial Informatics*, pp. 486–491. 2006.

148. A. Vijayaraghavan and D. Dornfeld. Automated energy monitoring of machine tools. In *CIRP Annals - Manufacturing Technology*, 59(1), pp. 21–24, 2010.

149. C.B. Vilakazi, T. Marwala, P. Mautla, and E. Moloto. On-Line Condition Monitoring using Computational Intelligence. In *Artificial Intelligence*, p. 23, 2007.

150. R.A. Virzi. Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough? In *Human Factors*, 34(4), pp. 457–468, 1992.

151. C. Wagner. Breaking the Knowledge Acquisition Bottleneck Through Conversational Knowledge Management. In *Information Resources Management Journal (IRMJ)*, 19(1), pp. 70–83, 2006.

152. W. Wahlster. Industrie 4.0: Vom Internet der Dinge zur vierten industriellen Revolution. In *Innovationskongress Märkte, Technologien, Strategien, Zentralverband Elektrotechnik- und Elektronikindustrie e.V.* 2011.

153. P. Wang, J.G. Zheng, L. Fu, E.W. Patton, T. Lebo, L. Ding, Q. Liu, J.S. Luciano, and D.L. Mcguinness. A Semantic Portal for Next Generation Monitoring Systems. In *Proceedings of International Conference on Semantic Web (ISWC)*. 2011.

154. G. Weidl. Object-Oriented Bayesian Networks for Condition Monitoring , Root Cause Analysis and Decision Support on Operation of Complex Continuous Processes : Methodology & Applications. In *Networks*, pp. 1–36, 2005.

155. T.M. Welte and H. Kile. Parameter estimation in degradation modelling: A case study using condition monitoring data from wood pole inspections. In *IEEE Trondheim PowerTech*, pp. 1–7. 2011.

156. P. Wennerberg, S. Zillner, M. Möller, P. Buitelaar, and M. Sintek. KEMM : A Knowledge Engineering Methodology in the Medical Domain. In *International Conference on Formal Ontology in Information Systems (FOIS)*. 2008.

157. A. Widodo and B.S. Yang. Support vector machine in machine condition monitoring and fault diagnosis. In *Mechanical Systems and Signal Processing*, 21(6), pp. 2560–2574, 2007.

158. H. Wiendahl, H. Elmaraghy, P. Nyhuis, M. Zah, N. Duffie, and M. Brieke. Changeable Manufacturing - Classification, Design and Operation. In *CIRP Annals - Manufacturing Technology*, 56(2), pp. 783–809, 2007.

159. Wolfgang Mahnke, S.H. Leitner, and M. Damm. OPC Unified Architecture Textbook. 2009.

160. L.R. Yang. Key practices, manufacturing capability and attainment of manufacturing goals: The perspective of project/engineer-to-order manufacturing. In *International Journal of Project Management*, 31(1), pp. 109–125, 2013.

161. M. Záková, F. Zelezný, and N. Lavra. Automating Knowledge Discovery Workflow Composition Through Ontology-Based Planning. In *IEEE Transactions on Automation Science and Engineering*, 8(2), pp. 253–264, 2011.

162. S. Zillner, A. Ebel, S. Martin, L. Abele, M. Gael, and N. Goldenberg. A semantic modelling approach for the steel production domain. In *Proceedings of European Steel Technology & Application Days*. 2014.