Resource Monitoring in Industrial Production with Knowledge-Based Models and Rules

Lisa Abele
Siemens AG
Corporate Technology
Munich, Germany
lisa.abele.ext@siemens.com

Martin Kleinstueuber
Department of Electrical Engineering
Technische Universität München
Munich, Germany
kleinsteuber@tum.de

Thorbjørn Hansen
Siemens AG
Corporate Technology
Munich, Germany
thorbjoern.hansen@siemens.com

ABSTRACT
The manufacturing domain currently experiences a significant increase in resource expenses for industrial plants. However, the implementation of systems to monitor the resource consumption in such complex plants requires high investment concerning time and manual effort. Our goal is to describe the plant by means of knowledge-based models and rules to implement a generic, semi-automated monitoring system which can be defined with lower initial effort and which can be adapted quickly to modifications. An advantage of this model-based approach is that the energy and resource consumption of each component in a plant can be associated with a sequence of operations and the effects on the overall system get visible. Another advantage of knowledge-based systems combined with rules is that they offer application independent solutions and flexibility. The paper outlines the state of the art of relevant technologies by describing several approaches, such as existing monitoring systems, rule engines and modeling tools. Furthermore, it describes a representative example that we will use in our further work to evaluate which tools are appropriate for a resource monitoring system.

Categories and Subject Descriptors
D.2.13 [Software]: Reusable Software—Domain engineering, Reusable libraries; I.2.4. [Computing Methodologies]: Knowledge Representation Formalisms and Methods—Representations (procedural and rule-based), Representation languages

General Terms
Design

Keywords
Knowledge Engineering, Industrial Production, Resources, Monitoring

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1. INTRODUCTION
Nowadays, manufacturing is part of all economic systems. The manufacturing plants are various and complex. In order to avoid downtime and reduce the resource costs (for energy, water, labor, gas, etc.) of a plant, not only subsequent analysis but also permanent monitoring of processes and components is needed. An appropriate method for addressing these issues is the implementation of resource monitoring systems in industrial production. Current solutions for monitoring systems in industrial production face several issues.

First of all, existing monitoring systems are, to the best of the author’s knowledge, neither reusable nor facility-independent and require significant manual investment at implementation. They are usually developed for one exclusive plant [1, 16] or industrial area only, for example spacecraft systems [11], electro-hydraulic linear drives [5] and others. Once implemented, these systems have to be redesigned frequently, because industrial plants regularly experience changes of design, technologies or business policies [24]. The development of a new plant or a modification of an existing plant require huge redesign effort to adapt the monitoring systems accordingly.

Another important aspect which is not considered in current solutions is reusability [10]. The advantage of reusing models is that one needs less manual effort to configure the monitoring system and the implementation can save time to a great extent. Existing monitoring systems are not reusing thresholds, working conditions or events defined in the manufacturing engineering chain, but are mainly configured and added manually. Currently, there is a big variety of engineering tools in the manufacturing engineering chain with proprietary or incompatible data exchange formats [22]. For this reason, there are efforts to introduce new data exchange formats such as the system-independent format CAEX [20], Semantic Web technologies [19] or other knowledge-based approaches [18] for the engineering of production systems. Their usage within automation systems is an interesting approach and has already been shown to be feasible [3]. However, there is not yet a standard facility-independent data format which is reusable, describes the static structure of a plant and its dynamic behavior as well.

A third issue is that monitoring depends on context, e.g. the current production schedule, the current point of time in the schedule, the state of components and the flow of control, materials, energy and information between components and its consequence on remote components or the entire plant. Besides, these components are often executing concurrent jobs. In a monitoring system, this leads to a combinatorial explosion in the number of states when multiple components and their multiple states are combined.
It is not viable to explicitly program the handling of the combined component states, instead rules are a suitable tool to cope with this problem. There is not yet a standard rule format for this task [21], but the Automation of Automation (AoA) approach [22] has investigated formalizing the interchange formats and the rules with regard to automation systems. Beyond that, rules are well-suited for an integration into knowledge bases, and rule-based approaches have already been successfully implemented in the industrial area [3, 9, 21]. But none of the recently developed rule-based systems aims at monitoring resources in a production plant.

In order to tackle the previously described issues, two steps are necessary. First, we will formalize knowledge about the plant in a model to make it processable for computer programs. Second, we will define rules which are applied to the knowledge stored in the plant model. As a result, it is possible to infer system states from the states of the individual components. By combining knowledge with rule-based representation methods a redesign of the monitoring system can be shortened and executed in a semi-automated manner. Another reason for choosing a rule-based approach is that in the production industry there are many similar assemblies which require similar monitoring solutions. In future research, we will investigate further which rule engine is best suited for our aims.

The scope of this paper is to describe a representative example that we will use for finding appropriate tools for resource monitoring in industrial plants. The method section of this paper consists of three parts. In the first part we discuss general requirements on the resource monitoring system. In the second part we examine the design of plant models with different tools. In the last part, we describe the sample plant and evaluate the efficacy and usability of the different approaches.

2. METHODS

Our intended research aim is to develop a resource monitoring system using knowledge-based models and rules to monitor a production facility. The three main elements needed for this purpose are a static model, a dynamic model and a logic for reasoning.

Figure 1 shows the general structure of our monitoring system. The building of the knowledge-based models starts at the design of a facility. The first step is the definition of a static and a dynamic model which include information on the plant components, their context, the structure of the entire facility and rules, from which a compiler generates a run-time model.

A monitoring run-time module executes this model. It reads and pre-processes the real-time data of the sensors, generates monitoring events and annotates them with a machine-readable semantic. In the last step the rule engine annotates the monitoring events with the states of the components and the system determined by applying the specified rules. These events can be shown to the user or be processed by further modules (e.g. a diagnosis module).

The first step of this thesis is to evaluate how to model the static and dynamic plant knowledge. At the beginning, we considered modeling approaches employed in software engineering and in industrial engineering. For the future system we will probably combine several of these approaches to obtain a complete monitoring system. For the first version of the simplified structure model of an industrial plant we chose an object oriented approach. Then, we built a generic system with Semantic Web technologies in Protégé. Later, we developed an example where the static plant model was built using the engineering tool CAEX and the corresponding dynamic model in PLCopen.

2.1 System requirements

Our initial question was what parameters in a plant have to be monitored to measure the energy consumption and how. When researching this question we decided to generalize it to monitor the resource consumption of a plant.

The total effort of a plant is influenced by the manufacturing process, the efficiency of machine components and the machine usage and control. In high speed machining, acceleration and deceleration forces are dominant and different machine concepts vary in energy efficiency due to way kinetic energy is fed back during deceleration, therefore acceleration sensors are an important element of our monitoring system.

Currently most of the industrial production plants are not monitoring their energy consumption [17] even if all the information necessary to measure the energy consumption would be available in their control systems [23]. Usually this information is not registered constantly but only in case of failure. The only exception are steel manufacturing facilities.

The most interesting components of a plant where energy consumption varies are filters, actuators, areas of heat exchange and components where electrolytic activities take place (e.g. semiconductors). Dependencies between process efficiency and major cutting parameters showed that energy consumption of a machine is dependent on process force in a nonlinear fashion [7]. Key figures, which are used for energy efficiency assessment, e.g. in the International Performance Measurement and Verification Protocol IPMVP are not detailed enough for production plants [6]. These research results showed us that it is important to take into account all secondary energy expenses in a production plant as well. This includes energy consumption of all machine components and supply units, power drain during idle times as well as other sources of energy such as pressured air, hydraulics, extracted air and coolants. For this reason, we decided that our system requires not only energy monitoring, but monitoring of all resources processed in a plant such as electric energy, water, heat, hydraulic fluid, lubricants, etc.

As a result, we deduce that for resource monitoring more sensors than in usual condition monitoring systems have to be inserted, such as acceleration sensors, flow rate sensors, ultrasonic sensors and else, to monitor the most important resources in a production plant.

During design the developer has to:

- Identify components and subcomponents of the plant
- Identify resources which have to be monitored
- Define relationships between components (including control, material, energy and information flow)
Several sequential steps are executed by the monitoring system, see Figure 2 for more details.

1. Capture sensor data
   - the Sensor measures one specific quantity at a certain time
   - the Sensor generates a raw signal with sensor specific values (e.g. 10 Volt)

2. Annotation of signal data
   - Continuous signal data is sampled into discrete properties
   - The quantity value of the signal data is enriched with quantity class, unit and timestamp

3. State identification
   - The Reasoner uses the numerical value of the property instance to determine the simple state of the component monitored by the sensor
   - The Reasoner infers a composite state from several simple states

Important requirements for the modeling approach are:

- the models are generic and facility-independent
- the notation is easy to use for all involved users (engineers, technicians, operators, etc.)
- the models are applicable to all research fields and industrial engineering areas
- reusability and shareability should be supported
- the models include components, their relationships and the context information of the plant

The most important criteria to guarantee a high reusability of models are:

- it must be possible to adapt the models to the application domain, particularly to define your own classes, roles, etc.
- support for libraries of classes, roles, components etc.
- it must be able to describe the context of the plant

### 3. MODELING APPROACHES

Our approach requires a combination of several approaches. Firstly, we need static models with classes, objects, roles, interfaces and attributes. Secondly, we need dynamic models to describe sequencing, concurrency, behavior and control aspects. To infer system states we need rules and a rule engine to execute them. We considered in the following four different approaches to meet our requirements:

- object-oriented modeling with UML
- ontologies with Semantic Web
- logical rules, for example SWRL and the rule engine Jena
- the industrial data format AutomationML with CAEX and PLCopen

#### 3.1 Object-oriented modeling with UML

As the first step in defining a structure model of production plants, we analyzed different production plants to extract a set of fundamental elements and described them in an object-oriented approach by defining classes and their relationships.

A typical plant is composed of several smaller stations (e.g. a welding station) which contains assemblies (e.g. a welding head) which consists of components (e.g. an electrode rotor) which may contain subcomponents in several levels. Usually the continuous measurement of data in industrial facilities is executed on component level. For this reason, the hierarchical model should reflect this composite structure. A usual structure of a plant hierarchy is represented in figure 3a. Since the hierarchies are application dependent, we generalize them as shown in figure 3b. We use the term component independently of the hierarchical level, and have a single relationship has_part.

In the next step, we generalize the model to allow an arbitrary number of relationships, e.g. connected_to and monitored_by. We also introduce interfaces as shown in figure 4. The most important relationships for our monitoring purpose are defined in this...
structure, but the number of relationships is not limited. A small example based on the predefined structure model is illustrated in figure 5 and describes the position of the component cauldron in the plant hierarchy. It is monitored by a temperature and a pressure sensor and connected to a valve. Figure 6 shows a typed model of the plant hierarchy with classes and objects in UML. Most CAE (Computer Aided Engineering) systems use types. The main benefits of typed models is reduced redundancy, type safety and a clear semantic of attributes and relations.

### 3.2 Domain model in Semantic Web

Our next step in order to build a domain specific knowledge model was the definition of an ontology based on Semantic Web technologies. To implement our application, we use Protégé which is a free, open source ontology editor and knowledge-based framework. We choose Protégé because it allows the creation of an ontology schema which can be exported easily to OWL and RDF/RDFS formats.

The advantage of ontologies is that you can build an extensible knowledge model. Compared to semi-formal or informal knowledge approaches which provide no logical formalism or model theory, the biggest benefit of ontologies is the automated validation and consistency checking [14].

The ontology schema consists of a set of predefined classes, attributes and properties. For our purposes, the most important contribution of an ontology is its ability to define classes with relationships and constraints among them. For example, the definition of domain and range for properties allow type inferences from predicates, for example the property monitors with domain Sensor and range Component, implies that sensors can only monitor components, this example is displayed in figure 7.

#### 3.2.1 Rules in Prolog and SWRL

Still, the model is not enough for our monitoring intent. Additionally, we need rules to extract information from the plant model and use sensor data to derive system states. Different rule engines support different semantics, for example Jena and Pellet support SWRL and Prolog supports Horn clauses.

In our first attempt, we examined the semantics of SWRL [15] (Semantic Web Rule Language) and Prolog to evaluate which functionalities we will need. Prolog is especially useful for real-time systems due to its good performance [25]. The rule engine Jena is embedded in Java and uses the rule format SWRL which combines
The rules are executed to calculate the state of an object depending on the input data at a given time. For example a rule “If the temperature of an object is over 100°C, then its state is set to TempTooHigh”.

The first drawback of SWRL in comparison to Prolog is that it does not support constructors. They are needed for composite states, for instance if several sensors are required to determine the state of a station. The following example in Prolog defines a composite state for a station using the constructor composite:

\[
\text{state(station, composite(Ps_st, Ts_st, As_st, Time);- state(ps, Ps_st,Time), state(ts, Ts_st, Time), state(as, As_st, Time).}
\]

This constructor defines a composite state of station at a certain time which consists of three individual states: the state of the pressure sensor (Ps_st), of the temperature sensor (Ps_st) and of the acceleration sensor (As_st). This cannot be expressed in SWRL.

The second drawback is that the integration of timestamps in SWRL semantics is complicated. SWRL only supports binary predicates, therefore two joins are required to insert a timestamp in a rule. Prolog supports n-ary predicate, so it’s possible to define the state \( s \) of a component \( c \) at a time \( t \) by stating

\[
\text{state(c, s, t)}
\]

In SWRL you have to rewrite this example with binary predicates and a “join variable” \( x \).

\[
\text{state(x, s) ∧ component(x, c) ∧ time(x, t)}
\]

The third drawback with the Semantic Web approach is that facts can only be created and not deleted. After a while this will lead to a huge, slow database including non-relevant facts. To resolve this issue, in our future research we will examine the use of forward and backward chaining and deletion of facts.

### 3.2.2 Modeling with AutomationML

As a second approach we examined if our requirements can be fulfilled by the domain-, application- and vendor-independent meta-data format for plant engineering AutomationML [12] which has been submitted as a national and international standard [8]. AutomationML supports several notations from different tools and disciplines, and supports different phases in the iterative plant engineering workflow covering different levels of detail [2]. We considered only two parts of AutomationML, CAEX for static models and PLCopen for dynamic models.

With CAEX you can describe the topology of a plant with properties and relations of objects in their hierarchical structure. With PLCopen you can describe sequences of actions, internal behavior of objects and I/O connections.

We will consider CAEX in more detail to define a general topology of industrial plants, including components, their attributes and relations (control, material, energy or information flows). The main part of the model is the plant hierarchy described by InstanceHierarchy and three libraries which define roles, interfaces and system units (see figure 8). Whereas InstanceHierarchy describes one single plant, InterfaceClassLib and RoleClassLib are libraries of interface classes and role classes that can be reused for many production plants. The library SystemUnitClassLib describes a set of devices that may be included in a plant and provide additional manufacturer-dependent information.

As mentioned, CAEX allows the user to store relationships between components. The modeling of relationships in CAEX is different from our first modeling approach (see figure 4), because the element Internal Link does not support different kinds of relationships. Also, in CAEX it is required to relate two component interfaces as shown in 9. With some additional effort, it is possible to express several kinds of relationships in CAEX by storing the relationship class in interfaces.

AutomationML suggests basic roles to model the structure of any industrial plant. Roles can be extended and define characteristic attributes of components. We have adapted them to our terminology in figure 10.

The results of our analysis of CAEX show that it meets most of our requirements, especially the reusability requirements. CAEX allows to adapt models to the application domain by defining roles and classes. Modeling of context information is possible using roles. You can use system unit classes, interface classes and role classes to store data that is common to several objects. Thus, common data can be centrally stored and exists only once, which guarantees high reusability.

Another important benefit of CAEX is that the domain-specific knowledge on causes and effects can be stored in roles. This knowledge is central in a monitoring system, because if you monitor certain effects you should be able to identify the root causes, e.g. with an additional diagnosis system.

However, CAEX suffers from some weaknesses. The CAEX
data is stored in an XML format with low readability. Another weakness is the definition of relationships as described above. Furthermore, the design of the plant topology is not restricted, thus dependent on the developer and may differ widely. This results in different classes representing the same devices. Most of the information on classes can be mapped into the other, but standardized CAEX classes would be a big benefit.

3.3 Comparison of CAEX and ontologies

On one hand, ontologies are more expressive than CAEX because they do not restrict the set of concepts like CAEX does. On the other hand, CAEX supports class attributes. If you try to express that in ontologies, you need instances, classes and meta-classes. The resulting ontology cannot be expressed in OWL DL, which seriously restricts reasoning on the ontology. We’ll consider OWL2 in future work.

3.4 Example: Oil production plant

Our preliminary results suggest that AutomationML is suited for the monitoring system. Therefore, we defined a first didactic example of an oil production plant with this approach. The task of this simple plant is to produce cooking oil from natural oil. The piping and instrumentation (P&I) diagram used in process industry is shown in figure 11. Several steps were necessary to construct a static as well as a dynamic model of this oil production plant. First, we designed the static structure model of the P&I diagram as illustrated in figure 11, then we expressed this structure model in CAEX and finally we built a dynamic model with OpenPLC consisting of a control and a monitoring system. AutomationML is able to represent the same information content as the graphical P&ID. One of the scopes of CAEX is the data exchange between P&ID tools.

The first design decision for defining a model in AutomationML is to choose a view on the model. Depending on where you want to place your emphasis, you can build your model in a component-oriented, process-oriented or product-oriented view. For our purpose we need to focus on the component-oriented view, because the monitored components are the central items in every plant. They execute processes and machine the products. In figure 12 the static model of the oil plant is shown with a clear distinction between the structure model and its context. We need the context model to express the dynamic behavior of the system. From this view, the model can be easily translated into a CAEX model.

3.4.1 Static model in CAEX

A main design decision of the static model is how to classify the objects. There are two possibilities, either roles or attributes can be used to classify objects, e.g. "renewable resources" can be defined as role or attribute of the element "water".

To measure the duration \( d \) of every process, the usage of timers is required. The modeling of time in different disciplines differs.

In mathematics, dependencies are expressed by functional equations, e.g. \( d(t_{\text{timer}}, \text{process}) = t_{\text{now}} - t_1 \). In software engineering a different notation is common: \( \text{process}.\text{timer}.d = t_{\text{now}} - t_1 \). CAEX does not support these notations, so we decided to model time duration in CAEX as attribute of the object "timer" and "process" as context by defining a new notation with "[.]" (see figure 13 for more details).

3.4.2 Dynamic Model in PLCopen

The manufacturing industry today usually uses systems for standardized automation devices, e.g. PLCs (Programmable Logic Controllers), which are programmed in IEC 61131-3 [13]. IEC 61131-3 defines several languages, which follow a function-oriented or procedural-imperative paradigm. AutomationML uses PLCopen to support several of the IEC standard like Gantt charts, PERT charts, impulse diagrams, state charts and Sequential Function Charts (SFCs), and stores them in the data format PLCopen XML.
For designing the dynamic model of our example we used SFCs, since it is an executable PLC program including a mapping to real control hardware. The control structure of the plant is required to describe general behavior and sequencing of the oil plant. Besides, we need to develop a monitoring part in SFC interacting with the usual control system. In this part we identify states by simple logical expressions in the monitoring sequencing.

We realized several monitoring concepts in SFC as illustrated in Table 1.

On the whole, this example shows that AutomationML is appropriate for our monitoring purposes. PLCopen offers the possibility to execute a dynamic model, identify simple alarm states and assign them to the plant components. One advantage is the simple implementation of a multitude of different aspects. Time aspects can for instance be modelled by an additional time qualifier, e.g., “DS #30min” and monitoring signals can be defined to perform continuous monitoring tests.

The languages of PLCopen are somewhat low-level, in our future work we will investigate how to generate it from more expressive models.

4. CONCLUSION

This thesis investigates which kind of knowledge representation and rule engine is feasible and appropriate for monitoring a production facility. In this first investigation, we evaluated the pros and cons of different modeling techniques and demonstrated in a small example how the monitoring system could be designed in the future.

Returning to the goal posed at the beginning of this study, we have shown on an example that knowledge-based systems and rules can be used for resource monitoring. By formulating and modeling this knowledge the engineer can use software tools so that the operator can monitor the resources consumed by the plant.

Our findings suggest that in general a knowledge-based plant model and the knowledge of the relation between facts can be formalized for computer evaluation. The monitoring knowledge can then be reused and applied elsewhere. Many plants in the same industrial production area, e.g., in process engineering, have the same component structure (e.g., a pump induces cool water in a cooling system surrounding a cauldron), the same typical problems (if the pump fails, the cooling system heats up) which cause the same typical effects (the pressure raises inside the cauldron, so the mixer in the cauldron fails as well).

This study suggests that knowledge-based models of components can be used to describe potential effects of deviations in sensor data. This knowledge can be applied in every plant containing the components to monitor it correctly. The main obstacle is to infer from the results of component monitoring how the overall system state changes. Every monitoring system has to meet the challenge to define an overall system state to determine whether a repair or modification is necessary. This approach reduces the complexity and improves the flexibility of a system.

Monitoring a plant will generate a huge amount of data. Therefore, we need to preprocess the data to avoid reasoning on the entire plant data. Methods of dimensionality reduction and compressive sensing can support us by tackling these issues. Otherwise we will certainly face data handling issues, additionally the run-time performance of the system increases.

This initial research has thrown up many questions in need of further investigation during my thesis. First, we plan to evaluate more methods and tools for rule execution. Second, we will continue the requirement analysis by interviews with plant engineers. Then, the implementation will be performed in a separate project at Siemens. Once the implementation is finished, we will test the approach on a realistic example. Finally we have to ensure that the models are generic for the industrial production domain.

5. ACKNOWLEDGEMENT

This work has been supported in parts by the German Research Foundation (DFG) funded Cluster of Excellence "CoTeSys" - Cognition for Technical Systems and by the Siemens AG - Corporate Technology.
6. REFERENCES

[10] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.
[17] J. Neidig. Interview with Dr. Jörg Neidig, Siemens AG.