Automated IT Management using Ontologies

Andreas Textor, Fabian Meyer, Reinhold Kroeger
Distributed Systems Lab
RheinMain University of Applied Sciences
Unter den Eichen 5, D-65195 Wiesbaden, Germany
{firstname.lastname}@hs-rm.de

***Abstract*—For the management of IT systems, numerous models, protocols and tools have been developed. To achieve the long-term goal of comprehensive, highly automated IT management, the various sources of information need to be combined. As syntactic translation is often not sufficient, ontologies can be used to unambiguously and comprehensively model IT environments including management rules. In this paper, we present an approach that combines the domain model, rules, instance data (which represents real-world systems) into an ontology. As the basis for an IT management ontology, we convert the Common Information Model (CIM), a Distributed Management Task Force (DMTF) standard, into an OWL (Web Ontology Language) ontology. Moreover, probabilistic knowledge of the domain is modeled using Bayesian networks and integrated into the ontology. Furthermore, the approach describes a runtime system that merges monitoring data into the ontology and then uses a reasoner to evaluate management rules.**

***Keywords*-ontology; IT management; CIM; Bayesian network**

I. Introduction

In the domain of IT management, numerous models, protocols and tools have been developed. Notable models include the OSI (Open Systems Interconnection) network management model (also known as Common Management Information Protocol (CMIP)) and the still widely used simple network management protocol (SNMP). A more recent approach to specify a comprehensive IT management model is the Common Information Model (CIM, [2]), a widely recognized Distributed Management Task Force (DMTF) standard. The more complex an IT environment gets, the more important the capability becomes to automate as many tasks as possible. Both commercial and free management tools and frameworks exist that cover different parts of the required feature set for management tasks, but usually not only a single tool, but a set of tools is used. In order to achieve a unified view of the heterogenous integrated management models, mappings between different types of models can be defined. However, syntactic translations are often not sufficient, when the same concept is represented differently in multiple domains. This problem can be approached by using ontologies to clearly define the semantics.

Only when a comprehensive formal representation of the domain data exists, that is also capable of modeling rules, a largely automatic management becomes possible, because then not only structural, but also behavioural information can be expressed in the model. To achieve such an automated management system, we describe a runtime system that imports the corresponding domain model into the ontology and evaluates the rules, based on up to date monitoring data from the system under management. In order to represent the monitoring data in the ontology, instance data is acquired at runtime and added to the ontology, so that rules can be evaluated by a reasoner according to both model and instance data.

The approach presented in this paper uses an OWL (Web Ontology Language, [3]) ontology to combine the domain model, instance data and rules defined in SWRL (Semantic Web Rule Language) in order to create a system that can automatically manage an IT environment. This results in a comprehensive knowledge base that includes both the statically loaded domain model and dynamically updated runtime information about the system under management, as well as rules to control the behavior of the system. To model entities and relationships of an IT environment, the CIM model was converted into an OWL ontology.

A domain as complex as IT management cannot be modeled solely using exact and complete information, which might not be available. Instead, probabilistic modeling and evaluation might be adequate. To enable that, the ontology and the runtime system need to be extended accordingly. As neither CIM nor an OWL ontology have native facilities for the representation of such information, Bayesian Networks are employed. Bayesian networks are probabilistic models to specify causal dependencies between random variables in a directed acyclic graph. To model probabilistic knowledge, ontology elements are annotated so that a Bayesian network can be partially derived from the ontology at runtime.

Section II gives a short introduction on the Common Information Model and describes related work in the context of ontologies and IT management. The concepts for the translation of CIM into an OWL ontology and the concepts for the combination of Bayesian networks with an ontology are described in Section III. Section IV gives an overview of our architecture for the runtime system, that is based on the aforementioned concepts for automated IT management. The paper draws a conclusion in Section V.

II. Related Work

A. The Common Information Model (CIM)

This section briefly describes the basic properties of the Common Information Model [2]. CIM is an object-oriented model that describes the entities in an IT environment and the relationships between them. This covers both hardware and software entities. The goal is to comprehensively model every aspect that is needed for consistently monitoring and managing the IT environment. CIM consists of three parts:

- A basic information model called the *meta schema*. The meta schema is defined using Unified Modeling Language (UML, [4]).
- A syntax for the description of management objects called the *Managed Object Format* (MOF).
- Two layers of generic management object classes called *Core Model* and *Common Model*.

Figure 1 shows the CIM meta schema definition in UML, from the CIM specification [2]. The meta schema specifies most of the elements that are common in object-oriented modeling, namely

- *Classes*, *Properties* and *Methods*. The class hierarchy supports single inheritance (generalization) and overloading of methods. For methods, the CIM schema specifies only the prototypes of methods, not the implementation.
- *References* are a special kind of property that point to classes.
- *Qualifiers* are used to set additional characteristics of Named Elements, e.g., possible access rules for properties (`READ`, `WRITE`), marking a property as a key for instances (using `Key`) or marking a class as one that can not be instantiated. Qualifiers can be compared to Java annotations; some qualifiers also have parameters.
- *Associations* are classes that are used to describe a relation between two classes. They usually contain two references.
- *Triggers* represent a state change (such as create, delete, update, or access) of a class instance, and update or access of a property.
- *Indications* are objects created as a result of a trigger. Instances of this class represent concrete events.
- *Schemas* group elements for administrative purposes (e.g., naming).

Properties, references, parameters and methods (method return values) have a data type. Datatypes that are supported by CIM include {s,u}int{8,16,32,64} (e.g., uint8 or sint32), real{32,64}, string, boolean, datetime, and strongly typed references (<classname> ref).

In addition to the CIM schema, CIM specifies a protocol, based on XML over HTTP, which is used by CIM-capable managers to query classes, instances and invoke methods against a so-called CIM object manager (CIMOM).

B. Ontologies in IT Management

There are several publications that examine the application of ontologies to the domain of IT management, e.g., [5], [6], [7]. The general consensus is that OWL is well suited for the modeling of IT systems, as it provides powerful modeling capabilities paired with the ability to formulate rules as part of the model and the ability to modularize the ontology. Still, both the complete translation or mapping of existing IT management models into OWL and the creation of an ontology-based automated IT management system are no solved problems.

In [5] the authors provide mappings for parts of different IT management models to OWL, including Structure of Management Information (SMI) and the Common Information Model (CIM). The resulting ontology can be used to combine the knowledge given in the different representations into a joint model. One problem the authors point out for the mapping is information that can be expressed in the original languages, but has no direct representation in OWL, such as the attachment of measurement units or access authorizations to properties. To solve this problem, the data is presented on the Resource Description Framework (RDF) layer of OWL. In RDF, it is possible to attach additional information to edges in the graph so that the data can be represented. However, this information is not available for evaluation by an OWL reasoner and therefore this approach has only limited use for automation that relies on the evaluation of rules from the ontology.

[6] describes how to represent several abstraction layers of a system in split ontologies to achieve a pyramid-like structure of ontologies, where often used ontologies are at the bottom of the figure. The reuse of components and models is an important topic in IT systems, and especially for ontology-based automation. The paper shows that OWL is capable of organizing several abstractions of a system in ontologies and reuse defined components in higher layers. This is an important aspect for the realization of a real-world management system.

A real-world management application is shown in [7] where ontologies are used to manage a network infrastructure. SWRL rules are used to create new object property connections between entities in case of a blackout. For this, properties and instance structures are observed. As a basis, Policy-based Network Management (PBNM) [8] was used. Rules are evaluated periodically during runtime, and new facts are added to the ontology. A management component observes the ontology and maps newly added facts to management operations to adjust the system.

In order to create a comprehensive ontology to model the system under management, a suitable domain model is required. The Common Information Model was examined in several publications (e.g., [9], [10]) and is often proposed as a domain model for the IT management domain, but the

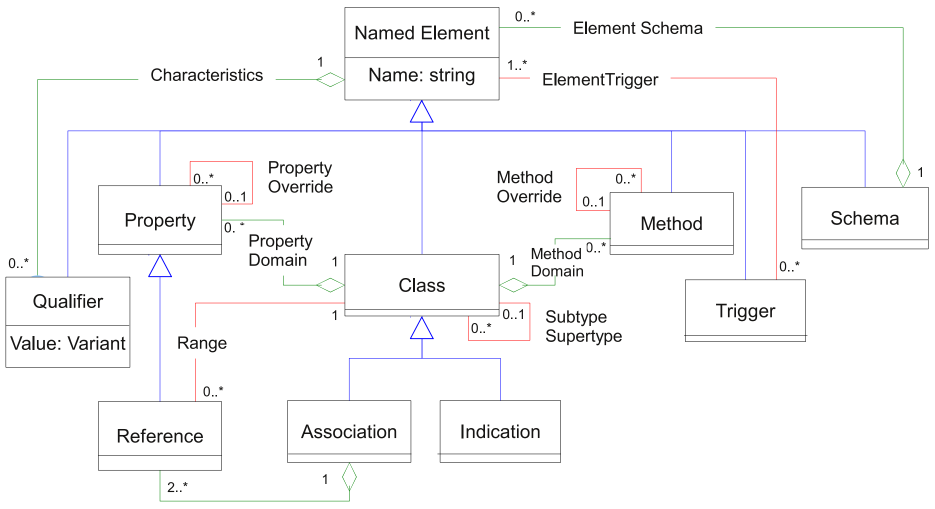

Figure 1. CIM Meta Schema [2]

authors in [9] show that it is a semi-formal ontology that has limited abilities for knowledge interoperability, knowledge aggregation and reasoning. In practice, this means that it is difficult to combine it with models from other domains, that it has no features for reasoning or the definition of rules as part of the model, and that it has only very limited built-in querying capabilities.

One solution to overcome the shortcomings of CIM while still benefiting from the comprehensive model is to translate CIM into a standard ontology format. In [9] the authors compare possible conversions of CIM to RDFS (Resource Description Framework Schema) and to OWL. They find that RDFS is unsuitable to express CIM as it does not allow to express constructs such as cardinality restrictions and some CIM qualifiers. In [5] the authors provide a possible mapping for a subset of CIM to OWL and the authors in [10] introduce a meta-ontology to model CIM constructs that have no direct OWL correspondence, but they do not describe how this meta-ontology is constructed, and their approach does not specify how several qualifiers and more complex elements, such as CIM methods, can be converted.

We developed a full mapping of CIM to OWL, which we use in this paper (see Section III-A) and which is first described in [11].

C. Bayesian Networks in IT Management

Bayesian Networks are used for the prediction of states for unobservable random variables. In IT systems management the probabilistic model of Bayesian Networkds suits root cause analysis and failure prediction well.

In [12] Bayesian Networks are used for hard disk failure prediction. Recorded Self-Monitoring, Analysis and Reporting Technology (SMART) data is used to generate the conditional probabilities for the nodes of the network. Two methods are used for the learning process, a clustering methods for sub model extraction and Expectation Maximization (see [13]) and Supervised naive Bayes Learning (see [14]). After the learning process, the SMART characteristics of a hard disk can be set as evidence in the network and a prediction for the probability of an upcoming failure can be made.

[15] considers the reliability for software systems. A special form of Bayesian Networks, the Markov Bayesian Networks, is used to predict failures in discrete time systems. Especially the working profiles of the system defined in [16] are considered as input parameters for the network. The conditional probabilities are extracted from software metrics. It is shown that the model does not provide optimal predications for failure rates, but better predictions than the Discrete Time Hyperexponential Model for Software Reliability [17].

[18] shows another attempt to predict software failures with Bayesian Networks. Complex internal relationships and correlations are considered to identify factors for failure contributions of components. For the construction of the network, a directed graph is generated where all variables are connected. Only boolean state spaces are allowed for variables. Every edge is weighted with its probability and a maximum weighted spanning tree is generated. By the choice of a root node a directed graph is achieved. Test data of the Eclipse Project [19] was used to show that the model can make statistically significant assertions about the failure probability.

D. Combination of Ontologies and Bayesian Networks in IT Management

There are no methods known to the authors for the combination of ontologies and Bayesian Networks in an IT Management context, but there are approaches to embed probabilities into OWL. In [20] the embedding of probabilistic knowledge for OWL class membership is presented. The major problems are the representation of probabilistic knowledge in OWL, the derivation of an acyclic graph and the construction of the conditional probabilities. Therefore, special OWL classes are defined to represent the expressions $P(A)$, $P(A|B)$ and $P(A|\overline{B})$, which have properties for conditions, values and probabilities. These properties are used to generate the conditional probabilities. A specially modified reasoner is needed to evaluate the ontology, so that existing reasoners cannot be used.

[21] defines the language PR-OWL. In contrast to [20], no existing language constructs of OWL are used, but a proprietary extension is defined. A probabilistic ontology must define at least one individual of the special class “MTheory”. A theory consists of a set of fragments that can be either context, residuum or input. Every random variable is a node with a set of possible values and conditional probabilities. With a special reasoner these constructs can be evaluated.

E. Belief Change

The process of changing beliefs to take into account a new piece of information about the world is called *belief change*. Belief change studies the process an agent has to perform to accommodate new or more reliable information that is possibly inconsistent with existing beliefs. Usually, beliefs are represented as a set of sentences of a logical language.

In the literature, three types of belief change operations can be distinguished: contraction, expansion and revision. Contraction is retracting a sentence from a belief set, expansion is adding a sentence to a belief set regardless of whether the resulting belief set is consistent or not, and revision is incorporating a sentence into a belief set while maintaining consistency. Alchourrón, Gärdenfors and Makinson [22] specified postulates for contraction and revision operators, which they claim should be satisfied by all rational belief change operators.

The problems described hold for the change of ontologies as well, as an ontology can be considered a belief base in the sense of the belief change theory. In this context, the problem is known as *ontology change*. Belief change theory can not be directly applied to description logics because it is based on assumptions that generally fail for description logics [23]. However, the authors in [24] show that all logics admit a contraction operator that satisfies the postulates except the recovery postulate. In [25], the authors show that the theory can be applied if the recovery postulate is slightly generalized.

Another approach to the problem of ontology update is taken in [26], which proposes an ontology update framework where ontology update specifications describe certain change patterns that can be performed. Change requests (adding or removing pieces of information to the ontology) are only accepted if the corresponding update specification accounts for it. The update specification is implemented similar to a database trigger and possibly carries out more ontology changes than explicitly requested to ensure ontology consistency.

Updating the ontology can be avoided, when changes over time are modeled in the ontology. For this approach, so-called fluents are used, where facts are tagged with the time or range of time at which they are valid. The authors in [27] describe how fluents can be modeled in OWL. However, this creates a large number of additional instances in the ontology, which makes it impractical for the application in the IT management context, where many changes of the ontology take place in short time frames.

III. Concepts

A. Transformation of the Common Information Model to OWL

As pointed out in Section II-B, a translation of CIM to OWL must be performed. The translation approach described in this section has first been published in [11] and is described in full detail in [28]. This section gives an overview of the translation approach and describes (previously unpublished) technical details necessary for the implementation of the translation using exemplary values. Note that the resulting OWL ontology is available for download at [29].

The translation creates an ontology that consists of two parts. The first part is a manually modeled meta-ontology that describes super classes, properties and annotations that meta-model CIM constructs, which can not be directly translated to OWL. The meta ontology has the namespace `cim-meta`. The second part is the CIM schema ontology, which is modeled using OWL-, RDFS- and CIM meta constructs, and which represents the actual CIM model. This part is generated programmatically by parsing the original MOF files and applying the following translation rules. The implementation uses pattern matching techniques on the abstract syntax tree of the CIM model to apply the translation rules.

Structural translation is mostly straightforward. CIM classes can be mapped to OWL classes, although a class in object-oriented modeling is not identical to the concept of a class in an ontology. Likewise, generalisation (inheritance) can be expressed using the OWL subclass concept `rdfs:subClassOf`. CIM has another basic construct for the expression of relationships, a so-called Association, which is a special kind of class with two typed reference properties (antecedent and dependent). Associations are mapped to OWL classes that inherit from the special meta class `cim-meta:CIM_Association`. CIM aggregations are handled accordingly.

Each CIM property is translated into an OWL object property and an OWL class that inherits from `cim-meta:CIM_Value`. The domain of the object property is the class that originally contained the property, while the range is the `CIM_Value` subclass. This class in turn then has a data property, which contains the actual value. The additional indirection is necessary for two reasons: CIM properties can have values of both primitive types such as `uint32` and references to classes, also the `CIM_Value` subclass is necessary to be able to express the CIM qualifiers on the property.

```
class CIM_System : CIM_EnabledLogicalElement {
  string Name;
}

class CIM_ComputerSystem : CIM_System {
  uint32 SetPowerState(uint32 PowerState, datetime
       Time);
}
```

Listing 1. CIM properties

In the following paragraphs, the mapping is illustrated using a concrete example. For the first class in Listing 1 the following OWL elements are created:

- An OWL class `CIM_System` that is a subclass of the OWL class `CIM_EnabledLogicalElement`
- An OWL class `CIM_System__Name_Value` that is a subclass of `cim-meta:CIM_Value`
- An OWL object property `CIM_System__Name` with domain `CIM_System` and range `CIM_System__Name_Value`

Information about qualifiers on the property can then be added to the `CIM_System__Name_Value` class as annotations or further object or data properties.

To translate methods, even more structural elements are required. The method itself, its parameters and return type, and the types of each parameter must be modeled. The second class in Listing 1 is translated into the following CIM elements:

- An OWL class `CIM_ComputerSystem` that is a subclass of the OWL class `CIM_System`
- An OWL instance `CIM_ComputerSystem__SetPowerState_Method` that is an instance of `cim-meta:CIM_Method` (which has data properties `cim-meta:methodName` and `cim-meta:methodType`)
- An OWL object property `CIM_ComputerSystem__SetPowerState` that has the domain `CIM_ComputerSystem` and the range `cim-meta:CIM_Method` and an annotation `cim-meta:methodInstance` that points to the instance
- An OWL object property `CIM_ComputerSystem__SetPowerState__Parameters` that has the method instance as domain and a range of an `owl:oneOf` enumeration
- The enumeration contains the instances `CIM_ComputerSystem__SetPowerState_Parameters_PowerState` and `CIM_ComputerSystem__SetPowerState_Parameters_Time`, which are both instances of `cim-meta:CIM_Method_Parameter`, which in turn has the data properties `cim-meta:parameterName`, `cim-meta:parameterType` and `cim-meta:parameterPosition`.

CIM datatypes are translated into the corresponding XSD datatype, as shown in table I.

Table I. Translation of CIM types to XSD types

CIM type	XSD type	CIM type	XSD type
uint8	unsignedByte	string	string
sint8	byte	boolean	boolean
uint16	unsignedShort	real32	float
sint16	short	real64	double
sint32	int	datetime	dateTime
uint32	unsignedInt	char16	string
sint64	long	uint64	unsignedLong

The translation of CIM qualifiers is performed by expressing the semantics of each qualifier using OWL features, as far as possible. In cases where this is not possible, corresponding classes and properties are modeled in the CIM meta ontology that the schema ontology can refer to. Table II gives an overview of the translation of CIM structures and qualifiers to OWL.

Table II. Translation of CIM constructs to OWL

CIM Construct	Translation in OWL
Abstract	cim-meta:isAbstract
Aggregate	Handled together with Aggregation
Aggregation	cim-meta:CIM_Aggregation, cim-meta:CIM_Aggregation_Parent, cim-meta:CIM_Aggregation_Child
Alias	owl:equivalentProperty
Association	cim-meta:CIM_Association, cim-meta:CIM_Association_Role
Class	owl:Class
ClassConstraint	cim-meta:classConstraint
Composition	cim-meta:CIM_Composition, cim-meta:CIM_Composition_Parent, cim-meta:CIM_Composition_Child
Correlatable	No translation
Datatypes	See table I
Default values	Union of original property range and default value singleton
Deprecated	owl:DeprecatedClass, owl:DeprecatedProperty
Description	rdfs:comment
DisplayName	cim-meta:displayName
Exception	cim-meta:exception

Experimental	`cim-meta:experimental`
In	`cim-meta:in`
Inheritance	`rdfs:subClassOf`
Key	`owl:inverseFunctionalProperty`
MappingStrings	`cim-meta:mappingStrings`
Max	`owl:maxCardinality`
MaxLen	`owl:Restriction, xsd:maxLength`
MaxValue	`owl:Restriction, xsd:maxInclusive`
Methods	`cim-meta:CIM_Method` (plus one instance), `cim-meta:CIM_Method_Parameter` (plus one instance), `cim-meta:methodName`, `cim-meta:methodType`, `cim-meta:parameterName`, `cim-meta:parameterType`, `cim-meta:parameterPosition`, object property for method, object property for method parameters
Method-Constraint	`cim-meta:methodConstraint`
Min	`owl:minCardinality`
MinLen	`owl:Restriction, xsd:minLength`
MinValue	`owl:Restriction, xsd:minInclusive`
ModelCorrespondence	`rdfs:seeAlso`
Out	`cim-meta:out`
Override	`rdfs:subPropertyOf`
Property	`cim:CIM_<Class>__<Property>_Value` (subclass of `cim-meta:CIM_Value`), `cim:CIM_<Class>__<Property>`
Property-Constraint	`cim-meta:propertyConstraint`
PUnit	`cim-meta:punit`
UMLPackage-Path	`cim-meta:UMLPackagePath`
Read	`cim-meta:readable`
Required	`owl:minCardinality` of 1
Terminal	`cim-meta:isTerminal`
Units	`cim-meta:units`
ValueMap	`cim-meta:valueMap`
Values	`cim-meta:value`
Version	`owl:versionInfo`
Write	`cim-meta:writable`

B. *Combination of Ontologies and Bayesian Networks*

The goal of the new architecture is to combine precise and probabilistic knowledge. A central concept is to model an ontology with entities and relationships and to derive the Bayesian Network dynamically from the ontology. Therefore, several specifications need to be defined to put some additional semantics into the ontology. Mainly,

- how random variables are represented in the ontology and
- how relationships are represented in the ontology.

1) Variable Representation: As mentioned before, OWL ontologies are able to represent continuous and discrete variables as data properties. As Bayesian Networks only operate on discrete random variables, a discretization must be applied. To discretize continuous variables, some additional information is needed. OWL does not support the adding of supplemental data to data property assertions. In [5], this problem was solved by adding the data on the RDF layer of OWL. The approach veers away from the concepts of OWL and the support for most editors and tools gets lost. Hence, for this approach another representation is used, which capsulates random variables through instances of variable classes, which has a data property that contains the actual value of the variable. There are three different types of variables:

- Continuous variables
- Discrete variables
- Enumerations

Continuous variables are containers for floating point values, discrete variables are containers for integers. In contrast enumerations do not store primitive data but OWL individuals as a value.

A mechanism is needed to map values of all three types of variables from the ontology to the generated Bayesian Network and back again. Since enumerations generally have just a small state space the values can be mapped one by one. For continuous and discrete variables the mapping is problematic and a discretization must be applied. A discretization for variables that are already discrete is needed, because the state space (in most cases full integer state space) is too large for Bayesian Networks. For a random variable x the size of its conditional probability table $probsize(x)$ grows with

$$probsize(x) = spacesize(x) \cdot \prod_{c \in cause(x)} spacesize(c) \quad (1)$$

where $cause(x)$ is the set of causing variables of variable x and $spacesize(c)$ is the size of the state space of the variable c.

To support the discretization mechanism, we defined special interval properties for both types of variables. These intervals are used to discretize values in the runtime environment. They are defined in a math-based syntax as additional data properties of the variable class. For the mapping from OWL to the network, the matching interval is taken. Every interval is an unique discrete value in the network. The mapping back from the network to OWL is more complicated, because the discrete data has to be enriched. For this, the median of the matching interval is used.

2) Relationship Representation: Another part of the OWL model are relationships. They describe the coherence between random variables of the system. Three types of relationships are considered in the model:

- Functional relationships
- Causal relationships
- Correlations

Functional relationships are based on known dependencies, which can be expressed by a mathematical formula. SWRL as part of the OWL specification already supports the usage of mathematical expressions and is used for the definition of functional relationships. The rules are evaluated at runtime by an OWL reasoner and new axioms are generated depending on the bound variables. Because correlations can be seen as bidirectional edges and causal relationships as unidirectional edges, the OWL object property concept can be used for the representation of both. In general it is not possible to connect data properties in OWL, but in this case it is feasible because all variables are already encapsulated by instances of the variable class.

3) Ontology Structure: A base ontology has been defined, which defined all OWL concepts needed for the use of the specialized variables and relationships in a client-specific ontology. Figure 2 shows the structure of the ontology, which is described in detail as follows.

- The `Variable` class is the base class for all variable types. It is the domain and the range for the `causation` object property, which defines causation between variables and the `correlation` object property that defines a correlation between variables.
- The `NumericVariable` class extends the variable class and is the base class for all numerical variables. It is the domain of the `interval` data property, which defines intervals for discretization and the the `unit` data property that defines the unit of a numerical variable.
- The `ContinuousVariable` class extends the numerical variable class and is used for continuous variables in the model. It is the domain of a `continuousValue` data property, which stores the discrete value of a continuous variable.
- The `DiscreteVariable` class extends the numerical variable class and is used for discrete variables in the model. It is the domain of a `discreteValue` data property, which stores the discrete value of a discrete variable.
- The `Enumeration` class extends the variable class and is used for enumerations in the model. It is the domain of the `enumerationClass` data property, which defines an OWL class as state space and the `enumerationValue` object property that stores the value of an enumeration variable.
- The `System` class, which is the base class for all root nodes of defined systems.

The ontology can be imported into any other ontology and instances of the classes and assertions of the properties can be created. To create a system model independent of these special properties and classes but capable of using the features it is possible to import both the system ontology and the variable and relationships defining ontology to a new ontology and define same individual, same object property and same data property axioms.

4) Joint Model Evaluation: For the evaluation of the relationships between variables different techniques are used. Since functional relationships are already defined as SWRL rules, the evaluation is simple. An OWL reasoner binds the variables in the body of each rule and creates the axioms in the head of the rule.

Causations are mapped to a Bayesian Network where each instance of the variable class becomes a node. For numerical variables each variable is checked for intervals. A discrete state is created for each interval in the state space of the node in the network. Enumerations are checked for their defined enumeration class and for each individual of this class a state is created with the unique name of the individual. Causal relationships between variables become arcs in the Bayesian Network.

After the structure of the network has been created the runtime values are mapped from the individuals in the ontology to the nodes in the network in every reasoning cycle. For a numerical variable the value is read, the fitting interval is found and the state of the node is set to the unique interval. For enumerations the individual is extracted and the state of the node in the network is set to the unique identifier of the individual.

In the next step an inference algorithm is applied to calculate the belief for the states of each unobserved variable (variables which have no value set in the ontology) which part a causal relationship. If the calculated belief is above a defined threshold the deduced value is fed back into the ontology as an property assertion for the variable. For enumerations that step is quite simple because the state is exactly the identifier of the individual. In case of numerical variables the matching interval is found and the median of the interval is set as value for the variable in the ontology. Values derived from the Bayesian Network are marked as being fuzzy by a property in the ontology so that other inference algorithms are aware of that fact.

Correlations are more complex to handle because they are based on precise knowledge (that the correlation exists) but the coherence between the variables is just a statistical measurement. An evaluation compared to that of causations is not possible, because correlations are bidirectional and thus cannot be presented in an acyclic graph. Therefore, correlations are analyzed offline for their functional relationship and are replaced by SWRL rules. These rules can be evaluated like the rules for functional relationships, the result is marked as being fuzzy as well.

Given that system facts can be revoked a mechanism is needed to revoke facts in the ontology as well as facts, which

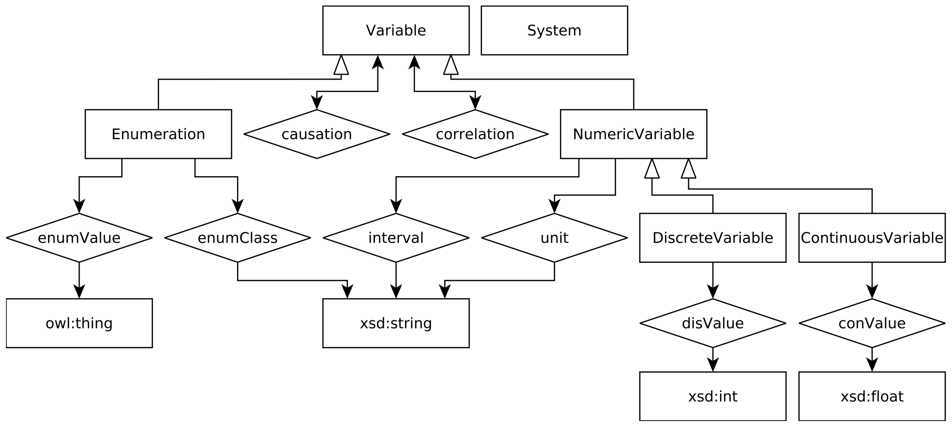

Figure 2. Entity structure of the created ontology.

have been derived based on those facts. OWL does not have a concept to automatically remove depending facts. Hence a directed acyclic graph is used to store the dependencies between the axioms of the ontology. If an axiom is removed, all child axioms are removed as well. Thus, not the whole derived knowledge must be revoked but only a subset of axioms.

IV. Architecture

A new architecture for ontology-based automated IT management is currently under development by the authors and the main ideas are sketched in this section. The architecture consists of a set of components (shown in Figure 3), which can be grouped into

- Importers that add new data to the ontology
- Reasoning components, which use the existing data to derive new knowledge
- Management components, which interact with the system under management.

The central element of the system is an ontology that is used as a shared knowledge base (blackboard) for all components. Each component can read data from the knowledge base and add or remove facts from it. Service invocations are used for the inter-component communication. The architecture is designed to be used in a distributed fashion.

A. Importers

The combination of different domain models raises the requirement for corresponding importers. These specific components know how to map the domain specific model to an ontology model. Hence, an interface is defined, which allows the use of new domain specific model importers. Implemented model importers are an ontology importer and a CIM importer. The ontology importer simply reads the data from an OWL ontology and adds the facts to the shared knowledge base. The CIM importer uses the mapping rules described in III-A to map the CIM schema to OWL facts.

As well as models, rules can be specified in a domain specific manner. Hence, an interface is provided for the implementation of domain specific rule importers. Internally, SWRL is used as rule format for the shared ontology and an according importer was implemented.

In general, the domain model contains just the taxonomy of the monitored system but not the instance data. Therefore, a component is needed that monitors the system under management and imports runtime data into the ontology by creating according instances. Such components are called instance importers. An interface is provided for the integration of domain specific instance importers. Already implemented instance importers are the log record importer, which maps log records to instances and relations, and the CIM instance importer, which uses the OpenPegasus CIMOM to get information from a CIM-based management system. Other application-specific instance importers can be added as needed.

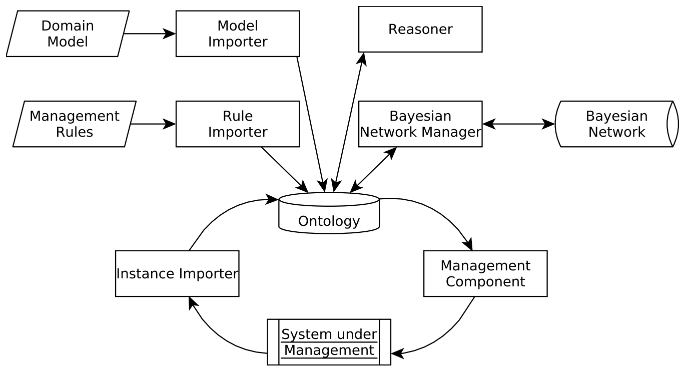

Figure 3. Components of the developed architecture

B. Reasoning

The strength of OWL and its formal grounding is the ability to reason new knowledge from an existing knowledge base. In our architecture this feature is used to derive new facts from the domain specific models, the imported rules and the monitored instance data.

In many cases it is insufficient to just consider exact knowledge in IT management, because side effects and complex relationships are either not known or can not be modeled in an adequate level of abstraction. But especially for state prediction and root cause analysis probabilistic knowledge and the statistical consideration of historical data are needed. Because of that, a concept is used to make probabilistic modeling and reasoning possible, which is described in detail in III-B. The structure of the Bayesian network is derived from the OWL model. The conditional probabilities are not modeled in the ontology directly, but trained using a maximum likelihood algorithm during a precedent training phase, which uses real data from the system under management.

In the next step the OWL model is analyzed for variable states, which will be set as evidences in the Bayesian network. Subsequently, an inference algorithm is applied to calculate the belief for the states of unobserved variables (variables which have no value set in the ontology). If the calculated belief is above a defined threshold, the deduced value is set for the variable in the ontology and can thereby be used by the exact reasoners for further reasoning. To ensure the knowledge exchange between the reasoning components a component can be called multiple times in a reasoning cycle.

C. Management components

Management components are used to reconfigure the system under management. They contain the knowledge that is needed to interact with a specific component of the system. Depending on the evaluation results of the rules, according actions are triggered. When CIM is used as a domain model, the management components can call methods on the CIMOM, which in turn controls the particular component, or it can execute external commands directly.

D. Runtime

The first step on application startup is the import of required domain models and rules using the according model and rule importers. After that, the management cycle is started (also known as MAPE-K loop [30], which stands for monitor, analyze, plan, execute and knowledge). The loop begins with the monitoring phase, where information from the system under management is read and imported into the ontology as instances.

In the analysis phase, the domain models, the rules and the monitored data are used for the reasoning of new knowledge. The reasoning process is shown in Figure 4.

The base ontology contains all the imported and monitored data. When the reasoning process starts, all data of the base ontology is copied into the working ontology. All reasoners are applied to this ontology sequentially and add

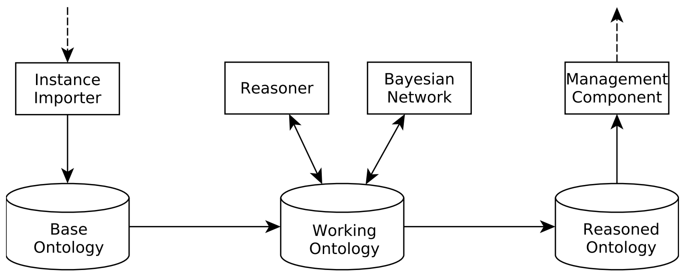

Figure 4. Multi step ontology reasoning process

their reasoned knowledge to it. When all reasoners have finished, the data of the working ontology is copied to the reasoned ontology, which is used for queries into the knowledge base and stays untouched until the next reasoning phase has finished.

The reasoning takes place in this multi-step process for two reasons: The first reason is handling ontology change, as new information can be added easily to an ontology, but not retracted easily. By keeping the base model and inferred knowledge from different reasoners in separate sub-ontologies, inferred knowledge from a single reasoner can be retracted without effort. The second reason is that the last version of the *reasoned ontology* can still be queried, while the new version is being created. As reasoning can be slow on large ontologies, this makes sure that clients do not block on queries but can always receive an instant reply. The query result therefore may be as old as one reasoning cycle.

The last steps in the cycle are the plan and execute phases. The management components use the data of the reasoned ontology to make management decisions and execute them on the system under management. The presented architecture was prototypically implemented in Java using the OSGi Framework as service middleware. The implementation comprises the core components and the specific features described above (i.e., model importers, rule importers and instance importers). More domain-specific components will be added in the course further application of the approach (see Section V).

For the service abstraction the interfaces of the OWL API are used.

V. Conclusion and Future Work

In this paper we presented an approach for ontology-based IT management. The approach comprises an architecture that uses an ontology which integrates the domain model, rules and dynamically updated instance data. Two main problems were solved: The first problem is the creation of a suitable domain model, which was covered by the translation of CIM to OWL and the expression of probabilistic knowledge using Bayesian networks. The integration of other domain models has yet to be examined. The second problem is the continuous update of the ontology with new facts. This is a topic of current research, and our solution is a multi-step reasoning process. Performance comparisons to other approaches and with different ontologies must be conducted.

Future work includes the development of importers for other domain models. The application of the presented approach is currently underway in two different concrete domains: One is the domain of storage management, the other is the domain of ambient assisted living (AAL). The application in these domains includes the development of domain-specific importers and the overall optimization of performance of the runtime system.

In the context of storage management the Storage Management Initiative Specification (SMI-S), which is a specialization of the CIM Model, can be used to manage storage systems. Rules, which are verbally defined in the specification, are formalized and integrated into the OWL model. Besides, the probabilistic part is used to make assertions about future states (e.g., how high is the probability of a full file system tomorrow if there is a peak) and to analyze previous scenarios (e.g., what was the most likely reason for a file server crash). In combination a pro-active management can be achieved and systems can be reconfigured before a failure occurs.

In the context of ambient assisted living the domain is a living environment, equipped with a set of sensors and effectors. That environment is modeled in a hierarchy of ontologies and monitored during runtime. The observed data is used to derive higher level knowledge, e.g., that lights should automatically be switched on or off when a person enters a room.

The proof-of-concept implementation of the ontology-based management system and the integration of probabilistic knowledge with the OWL ontology enables rule-based automatic management of domains for which a domain ontology was created.

References

[1] A. Textor, F. Meyer, and R. Kroeger, "Semantic Processing in IT Management," in *Proceedings of the Fifth International Conference on Advances in Semantic Processing (SEMAPRO)*, Lisbon, Portugal, November 2011.

[2] Distributed Management Task Force, "Common Information Model (CIM)," http://www.dmtf.org/standards/cim/, 2012-12-18.

[3] World Wide Web Consortium, "OWL Web Ontology Language," http://www.w3.org/TR/owl2-overview/, 2012-12-18.

[4] Object Management Group, "Unified Modeling Language (UML)," http://uml.org/, 2012-12-18.

[5] J. E. L. De Vergara, V. A. Villagrá, and J. Berrocal, "Applying the Web ontology language to management information definitions," *IEEE Communications Magazine*, vol. 42, no. 7, pp. 68–74, July 2004.

[6] J. E. L. De Vergara, A. Guerrero, V. A. Villagrá, and J. Berrocal, "Ontology-Based Network Management: Study Cases and Lessons Learned," *Journal of Network and Systems Management*, vol. 17, no. 3, pp. 234–254, September 2009.

[7] A. Guerrero, V. A. Villagrá, J. E. L. de Vergara, A. Sánchez-Macián, and J. Berrocal, "Ontology-Based Policy Refinement Using SWRL Rules for Management Information Definitions in OWL," *Large Scale Management of Distributed Systems*, vol. 4269, pp. 227–232, October 2006.

[8] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, "Terminology for policy-based management." The Internet Society, November 2001.

[9] S. Quirolgico, P. Assis, A. Westerinen, M. Baskey, and E. Stokes, "Toward a Formal Common Information Model Ontology," pp. 11–21, November 2004.

[10] M. Majewska, B. Kryza, and J. Kitowski, *Translation of Common Information Model to Web Ontology Language*, ser. Lecture Notes in Computer Science. Berlin: Springer Berlin Heidelberg, May 2007, vol. 4487, pp. 414–417.

[11] A. Textor, J. Stynes, and R. Kroeger, "Transformation of the Common Information Model to OWL," in *10th International Conference on Web Engineering - ICWE 2010 Workshops*, ser. LNCS, vol. 6385. Springer Verlag, July 2010, pp. 163–174.

[12] G. Hamerly and C. Elkan, "Bayesian approaches to failure prediction for disk drives," in *In Proceedings of the eighteenth international conference on machine learning*. Morgan Kaufmann, July 2001, pp. 202–209.

[13] T. M. Mitchell, *Machine Learning*, 1st ed. McGraw-Hill Science/Engineering/Math, March 1997.

[14] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, September 2006.

[15] C.-G. Bai, "Bayesian network based software reliability prediction with an operational profile," *J. Syst. Softw.*, vol. 77, no. 2, pp. 103–112, August 2005.

[16] J. D. Musa, "Operational profiles in software-reliability engineering," *IEEE Software*, vol. 10, pp. 14–32, March 1993.

[17] M. Kaaniche and K. Kanoun, "The discrete time hyperexponential model for software reliability growth evaluation," in *Software Reliability Engineering, 1992. Proceedings., Third International Symposium on*, October 1992, pp. 64 –75.

[18] Y. Liu, W. P. Cheah, B.-K. Kim, and H. Park, "Predict software failure-prone by learning bayesian network," *International Journal of Advanced Science and Technology*, vol. 35, 2006.

[19] Eclipse Foundation, "The Eclipse Project," http://www.eclipse.org/, 2012-12-18.

[20] Z. Ding and Y. Peng, "A probabilistic extension to ontology language owl," in *In Proceedings of the 37th Hawaii International Conference On System Sciences (HICSS-37), Big Island*, January 2004.

[21] P. Cesar, G. Costa, K. B. Laskey, and K. J. Laskey, "Pr-owl: A bayesian ontology language for the semantic web," in *Center for Technology-Enhanced Learning, University of Missouri-Rolla*, 2003.

[22] C. E. Alchourrón, P. Gärdenfors, and D. Makinson, "On the logic of theory change: Partial meet contraction and revision functions," *The Journal of Symbolic Logic*, vol. 50, pp. 510 – 530, June 1985.

[23] G. Qi and F. Yang, "A survey of revision approaches in description logics," in *Web Reasoning and Rule Systems*, ser. LNCS, vol. 5341. Springer, November 2008, pp. 74–88.

[24] G. Flouris, D. Plexousakis, and G. Antoniou, "AGM Postulates in Arbitrary Logics: Initial Results and Applications," 2004.

[25] M. M. Ribeiro and R. Wassermann, "First steps towards revising ontologies," in *Proc. of WONRO'2006*, 2006.

[26] U. Lösch, S. Rudolph, D. Vrandečić, and R. Studer, "Tempus fugit - towards an ontology update language," in *6th European Semantic Web Conference (ESWC 09)*, vol. 1. Springer, January 2009, pp. 278–292.

[27] C. Welty, R. Fikes, S. Makarios, C. Welty, R. Fikes, and S. Makarios, "A reusable ontology for fluents in owl," in *In Proceedings of Formal Ontology in Information Systems (FOIS)*, November 2006, pp. 226–236.

[28] A. Textor, "Semi-Automatic Management of Knowledge Bases Using Formal ontologies," Master's thesis, Cork Institute of Technology, Ireland, March 2011.

[29] RheinMain University of Applied Sciences, Distributed Systems Lab, "CIM OWL Ontology," http://wwwvs.cs.hs-rm.de/oss/cimowl/index.html, 2012-12-18.

[30] IBM Corporation, "An Architectural Blueprint for Autonomic Computing, Technical Whitepaper (Fourth Edition)," June 2006.