IT Management Using a Heavyweight CIM Ontology

Andreas Textor[1,2], Jeanne Stynes[2], and Reinhold Kroeger[1]

1 **RheinMain University of Applied Sciences**
Distributed Systems Lab
Kurt-Schumacher-Ringer 18, D-65197 Wiesbaden, Germany
{andreas.textor,reinhold.kroeger}@hs-rm.de

2 **Cork Institute of Technology**
Department of Computing
Rossa Avenue, Bishopstown, Cork, Ireland
jeanne.stynes@cit.ie

Abstract

This paper presents an approach for ontology-based IT management based on a heavyweight (formal) ontology using the Web Ontology Language (OWL). The ontology comprises a complete OWL representation of the Common Information Model (CIM) and management rules defined in the Semantic Web Rule Language (SWRL). The ontology not only models the managed system types, but a runtime system dynamically updates model instances in the ontology that reflect values of managed system entities. This allows the evalution of rules that take into account both model and model instances. A reaction module uses the CIM interface of the managed system to invoke CIM methods according to rule evaluation results, thus resulting in automated management. In order to ensure the consistency of the ontology when changes are performed, belief change theory is employed.

1998 ACM Subject Classification C.2.3 Network Operations

Keywords and phrases CIM, OWL, ontology, SWRL, management

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.73

1 Introduction

Knowledge bases grow in size and complexity in every domain. Regardless of the concrete field, the number of data sources and formats that are used in each knowledge base increases constantly. A common goal in this respect is to allow interoperability between different knowledge bases and integration and aggregation of data, as well as the possibility to perform automated reasoning. Established domain models can be used to create a unified view on the knowledge base, but they often have major shortcomings to be able to be used for comprehensive management of the knowledge base: They often have limited support for knowledge interoperability and aggregation, as well as reasoning, and constraints that are part of the models are often expressed using natural language, which renders them less useful for automatic evaluation. Also, the models usually do not deal with instance data that is acquired from the system which the model represents. Unless instance data is kept in a uniform way together with the model, the definition of constraints or rules that affect both model and model instances is not possible.

A formal ontology is a type of knowledge base that can help resolve these issues: While a *lightweight ontology* defines a hierarchy of concepts and a hierarchy of relations, i.e. a

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 73–84

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

description of the concepts and relationships that exist in a domain, a *heavyweight ontology* or *formal ontology* (the terms are often used interchangeably [24]) can contain domain-specific information models, behaviour rules, constraints and axioms, and can also contain the corresponding model instances.

Management of an IT environment is one example of a domain for which management using a formal ontology is desirable. Over time, a number of commercial and free systems for comprehensive management of IT environments have been developed. Usually, the management system is not comprised of a single tool, but consists of a set of different tools. To provide a unified view on the environment and to allow interoperability between multiple management tools and managed elements, several integrated network management models were developed. Notable examples are the OSI network management model (also known as CMIP, the name of its protocol) 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) which is described in more detail in section 2.1. This widely recognized Distributed Management Task Force (DMTF) standard allows consistent management of managed elements in an IT environment. CIM is actively used; for example, the storage management standard SMI-S (Storage Management Initiative Specification) of the SNIA (Storage Networking Industry Association, [19]) is based on CIM.

CIM can be used to create a unified view on the managed system, but suffers from the shortcomings mentioned earlier: It is a *semi-formal* ontology [4, 18] that only has limited abilities for knowledge interoperability, knowledge aggregation and reasoning [16]. To create a formal IT management ontology, two parts are required; the first part being a comprehensive domain model in a suitable format. The second part is a runtime system that administers the domain model and model instances that represent entities of the managed environment. As the runtime system provides means to add and delete model instances to reflect changes in the managed environment, it must take care that adding and deleting facts may not render the knowledge base inconsistent.

In this paper we present an approach for building a formal ontology for IT management based on CIM including a domain model, model instances and rules, a runtime system that manages the ontology, interfaces with the managed system to add and delete model instances, and an engine to evaluate management rules. For the domain model, a conversion of CIM to the *Web Ontology Language* (OWL) is performed. In order to assure the consistency of the knowledge base, *belief change theory* is employed. Section 2 gives an overview of CIM and belief change theory as background, while section 3 describes related work. Section 4 explains the management approach and how the runtime system is constructed. In section 5, implementation details and performance measurements of the approach are discussed. The paper closes with a summary in section 6.

2 Background

2.1 DMTF CIM Overview

This section briefly describes the basic properties of the Common Information Model. CIM defines how managed elements in an IT environment are represented as a set of objects and relationships between them. The model is intended to allow a consistent management of these managed elements, independent of their manufacturer or provider. Unlike SMI (Structure of Management Information, the model underlying the popular SNMP protocol), CIM is an object-oriented model. CIM consists of

- A basic information model called the *meta schema*. The meta schema is defined using the Unified Modeling Language (UML, [13]).
- 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, as shown in the CIM specification [6]. The meta schema specifies most of the elements that are common in object-oriented modelling, 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).

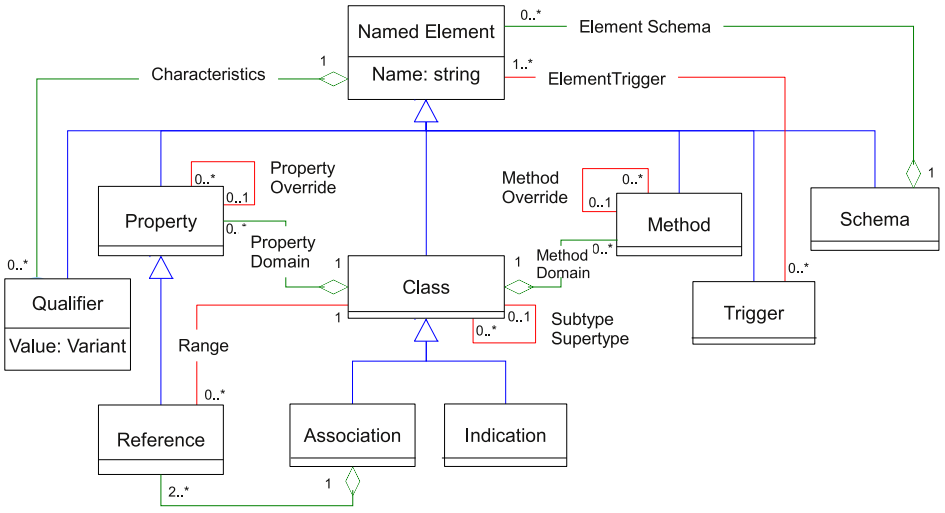


Figure 1 CIM Meta Schema

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 network managers to query classes, instances and invoke methods against a so-called CIM object manager (CIMOM).

2.2 Belief change and ontology 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. Most formal studies on belief change are based on the work of Alchourrón, Gärdenfors and Makinson (AGM) [2, 1], which is now commonly referred to as the *AGM theory*. The AGM theory distinguishes three types of belief change operations that can be made: 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, and revision is incorporating a sentence into a belief set while maintaining consistency. The AGM trio specified postulates for contraction and revision operators which they claim should be satisfied by all rational belief change operators.

As an ontology can be considered a belief base in the sense of the belief change theory, the problems described hold for the change of ontologies as well. In this context, the problem is known as *ontology change*. Belief change theory can not be directly applied to description logics (which are the theoretical foundation of OWL) because it is based on assumptions that generally fail for description logics [15]. However, the authors in [7] show that all logics admit a contraction operator that satisfies the AGM postulates except the recovery postulate. In [17], the authors show that the theory can be applied if the recovery postulate is slightly generalized and to achieve this, replace the recovery postulate with the relevance postulate by Hansson [9].

Another approach to the problem of ontology update is taken in [11], which proposes an ontology update framework where ontology update specifications describe certain change patterns that can be performed. An agent (knowledge worker) "issues a change request by providing a piece of knowledge to be added to or deleted from the ontology". The change request is only accepted when the given ontology update specification accounts for it. An applicable update rule from the update specification is determined and similar to a database trigger, "the change request is acted on accordingly by denying or accepting it but possibly also by carrying out more ontology changes than explicitly requested."

3 Related Work

The idea of applying semantic web technologies to the domain of IT management has been examined in several publications, notably by De Vergara et al., e.g. [3, 8], where a semantic management framework is presented which integrates different definitions of the managed resources, taking into account the semantic aspects of those definitions. In later works they favor the use of OWL for the definition of the management ontology. As a domain model, CIM is often proposed, although the conversion of the native format in which CIM is specified into an ontology is not trivial. In [5] the authors provide a possible mapping for a subset of CIM to OWL and in [16] 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. Instead, they construct an OWL-based ontology for CIM by using a previously defined mapping of UML. Most CIM concepts (e.g. CIM qualifiers) however, have no mappings to either UML or OWL constructs, so the resulting ontology lacks a large part of the information that is provided in the original model. The authors in [12] 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. The approach presented in this paper thus relies on the translation of CIM to OWL described in section 4.2, and in more detail in [20].

To be able to model a heavy-weight ontology for management purposes, several papers examine how SWRL (Semantic Web Rule Language) can be employed. In [8], the authors propose a formal definition of the different management behaviour specifications integrated with the management information definitions. They try to include both behaviour definitions included implicitly in the management information and explicitly in policy definitions in the resulting management ontology. Other works, notably [23] and [22], define a configuration management ontology and augment the ontology with behaviour definitions in SWRL and service definitions in OWL-S (an OWL ontology for describing Web Services).

4 Approach

4.1 Architecture

The goal of the approach presented in this paper is the management of a system using a suitable domain ontology and a runtime system that performs the actual management using the ontology. Although the approach is not limited to IT management (as the domain ontology and interfacing components can be replaced), in this paper it is applied to the domain of IT management. The domain ontology is defined in OWL, as this format is the de-facto standard ontology language and allows the description of ontologies using classes, object properties, data properties, cardinalities etc. As described in section 3, the Common Information Model (CIM) has been examined and found to be an appropriate model for an IT management ontology, however the translation of CIM to OWL has to be performed first.

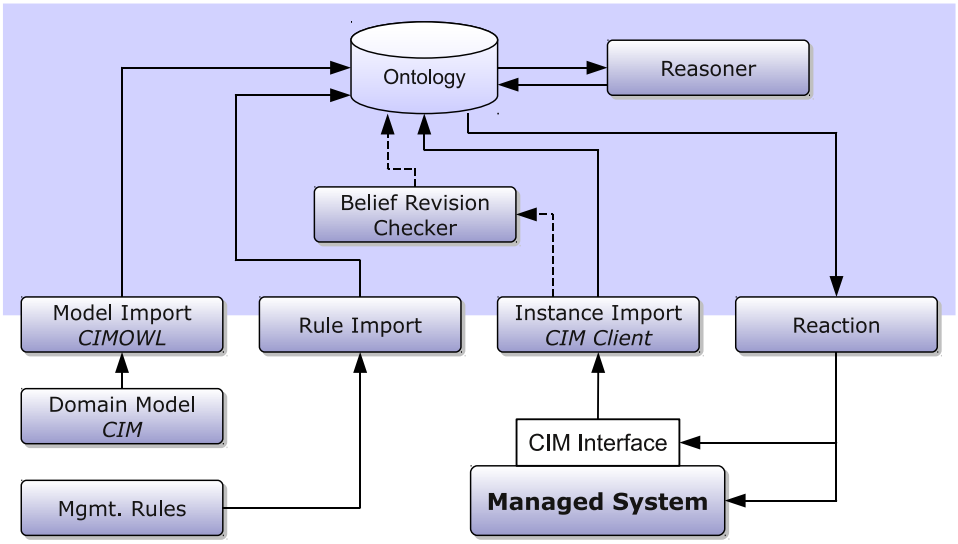


Figure 2 Architecture

Figure 2 shows an overview of the proposed architecture. The grey box at the top represents the runtime system and its components, while the elements at the bottom of the diagram show the parts the runtime system interacts with. The components of the runtime system are described in detail in the following sections.

4.2 Model Import

First of all, the runtime system has a component to import the domain ontology into the runtime ontology (which is shown at the top of figure 2). The domain ontology is a static model that describes the entities in the managed system in detail. This component is responsible for the translation of CIM to OWL. As described in section 3, this translation is not trivial if most or all CIM constructs are to be translated, including CIM qualifiers. When CIM is translated into OWL, the result is named *CIMOWL* here.

Some elements from the CIM schema can be translated in a straightforward way, while other elements, especially qualifiers, can not be directly translated. To solve this problem, the translation approach used here consists of two parts: One part is a CIM meta-ontology, which is statically modelled (i.e. manually) and which consists of super classes, properties and annotations that meta-model CIM constructs which can not be directly translated to OWL. The second part is the CIM schema ontology, which is modelled using OWL, RDFS and CIM meta constructs, and which represents the actual CIM model. It is automatically created by parsing and translating the MOF representation of the CIM schema. One particular part of the translation, the implementation of CIM properties in the OWL model, is given here as an example.

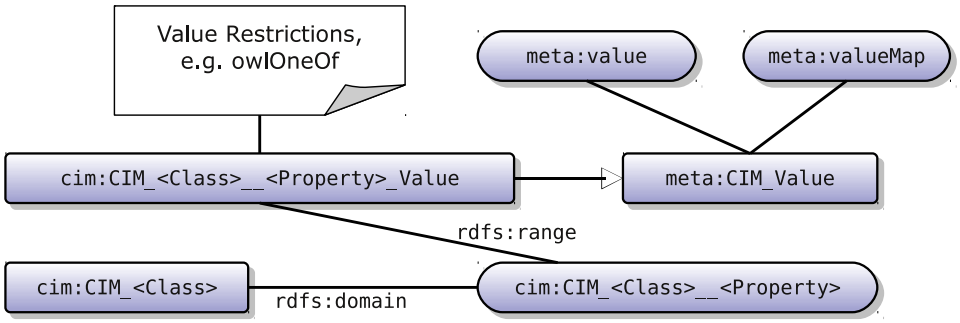


Figure 3 Modelling of properties

To implement CIM properties in the OWL model, a structure as shown in figure 3 is used. In the figure, rectangular boxes represent OWL classes, rounded boxes with one connection represent datatype properties and rounded boxes with two connections represent object properties. Each CIM property is modelled as a combination of an OWL object property and an OWL class. This class inherits from the meta ontology class `CIM_Value` that has two datatype properties, "`value`" and "`valueMap`". To model the semantics from the original `ValueMap` and `Values` qualifiers, the class is restricted using `owl:oneOf` to allow only instances from a list of `CIM_Value` instances (one for each `ValueMap-Values` combination).

The translation of CIM to OWL is described in more detail in [20]. Only constructs that are valid in OWL DL are used; the resulting ontology has the expressiveness of OWL DL, which allows for decidable reasoning.

The translation of the CIM schema has to be performed only once (and whenever the ontology has to be updated, which can be necessary if a new version of the CIM schema is released). CIM schema version 2.23.0 consists of about 1400 MOF files, containing roughly the same amount of OWL classes, and is converted to about 70,000 OWL axioms. The translation result is an OWL file which can be loaded by the runtime system on startup.

4.3 Instance Import

The instance import component is the component of the runtime system reponsible for acquiring up-to-date information from the managed system and incorporating it into the runtime ontology. As such, it has three concerns: The first task is that of a CIM client. As CIM is the domain model of choice, any CIMOM (CIM Object Manager) can be queried for CIM instances. These instances represent concrete entities of the managed system, such as hardware or software components. Queries are executed using the CIM protocol and result in an XML document that describes a collection of instances.

The second task of the instance import component is the conversion of this CIM instance data into the corresponding OWL instances. When performing this conversion, the resulting OWL instances have to be compliant with the OWL representation of the CIM schema that was created in the model import component and described in section 4.2. This means that each CIM instance is mapped to a collection of OWL axioms (for class instances, object properties and datatype properties, as necessary) that represent the same concept. Also, primitive values must be converted accordingly: Each signed and unsigned number type is translated to its XSD equivalent, e.g. `uint8` becomes `xsd:unsignedByte`, `sint32` becomes `xsd:int`, `real32` becomes `xsd:float`, and so on. Translation is mostly straightforward, except for `char16`, which has to be translated into a string, and `datetime`, which has a corresponding XSD data type but which specifies a different lexical representation for the datetime string.

The third task for the import component is to act as a revision operator in the sense of belief change theory (cp. section 2.2). When OWL axioms that are created when reading CIM data from the managed system are to be added to the runtime ontology, they must not render the ontology inconsistent. As the import component normally does not create axioms that perform structural changes on the ontology when added, but only adds or changes values, most of the problems that are studied in the field of ontology change can be circumvented, i.e. a change that leads to an inconsistency does not happen easily. The component must ensure however, that adding values that represent the current state of some part of the managed system does not lead to ambiguities. When an instance for an updated value is to be added, instances of the class the value belongs to have to be removed accordingly. The import component has to make sure at this moment that the adding and removing operations, once performed, result in a consistent state of the ontology, and in unambiguous information.

4.4 Belief Revision Checker

The import component implements an interface for belief revision checks. AGM postulates, updated for ontology change (see section 2.2), are implemented by the belief revision checker component. As it is not possible to create a generic belief revision operator that works for any knowledge base and any new fact, the belief revision checker implements the checks necessary for a given revision operator to verify that it keeps the ontology consistent. Because these checks have a comparably high computational complexity for reasoning, which can slow down the runtime system considerably, the belief revision checker can be switched on and off as necessary. When a new revision operator is installed in the system, the system is run with the revision checker switched on, using representative data, to verify that the operator works as intended (i.e., testing every case of axioms the revision operator adds or updates in the ontology). For production use, the revision checker is then switched off, which reduces the overhead to a minimum (the complexity for adding or removing instances in the ontology).

Listing 1 Sample SWRL rule

```
CIM_DataFile(?f),
CIM_DataFile(?reffile),
CIM_LogicalFile__LastAccessed(?f, ?lastaccvalue),
CIM_LogicalFile__LastAccessed(?reffile, ?reflastaccvalue),
CIM_LogicalFile__Name(?f, ?filenamevalue),
CIM_LogicalFile__Name(?reffile, ?reffilenamevalue),
CIM_LogicalFile__FileSize(?f, ?filesizevalue),
value(?reffilenamevalue, ?reffilename),
endsWith(?reffilename, "/var/reffile"),
value(?filesizevalue, ?size),
value(?lastaccvalue, ?lastacc),
value(?reflastaccvalue, ?reflastacc),
greaterThan(?size, 150000),
lessThan(?reflastacc, ?lastacc) ->
LargeOldFile(?filenamevalue)
```

The instance import component can now continue its work, as its functionality has been verified.

4.5 Specification and Import of Rules

While the model import component provides the static domain model in the ontology, and the instance import component dynamically adds and removes model instances to reflect changes in the managed system, the rule import component loads rules from external OWL files into the runtime ontology. Rules are defined using the Semantic Web Rule Language (SWRL) which can be embedded in an OWL ontology and can take into account both the statically loaded CIM model and the dynamically added model instances. Therefore it is possible to define rules that are triggered depending on the current state of the managed system.

Each set of rules for a certain check or task is accompanied by the definition of classes that are used in the context of these rules to denote the result of the triggered rule. Listing 1 shows a sample SWRL rule, which checks for files that surpass a certain file size (150,000 bytes in this case) and are older than a reference file (`/var/reffile`). As CIM does not natively support the definition of rules, no standard ruleset exists that could be used for testing the system. The sample rule given here was thus devised for this purpose.

`CIM_DataFile(?f)` identifies an instance `f` as belonging to the `CIM_DataFile` class, while `CIM_LogicalFile__FileSize(?f, *)` match for the `FileSize` and `Name` properties of the file, respectively. `CIM_LogicalFile__LastAccessed` matches for the dateTime value at which the file was last accessed. The expression `value(?filesizevalue, ?size)` extracts the numerical value from the `CIM_Value` datatype property that is attached to the file size object property. Using the SWRL builtin `swrlb:lessThan` on the values of the `LastAccessed` fields, it can be determined if the file in question is older than the reference file.

While the classes that start with `CIM_*` are defined in the CIM ontology, the class `LargeOldFile` is defined to accompany the rule. All instances of `CIM_DataFile` that have a `CIM_LogicalFile__FileSize` attribute with a value greater or equal 150,000 and were last accessed before the reference file are defined to also belong to the class `LargeOldFile`. The rule can be extended and used in a use case where large files that are older than a

certain date on a filesystem should be compressed or archived. This example shows the basic approach for the definition of rules. CIM models not only the file system but other aspects of an IT environment as well, so rules covering other areas can be defined similarly.

4.6 Reasoner and Reaction

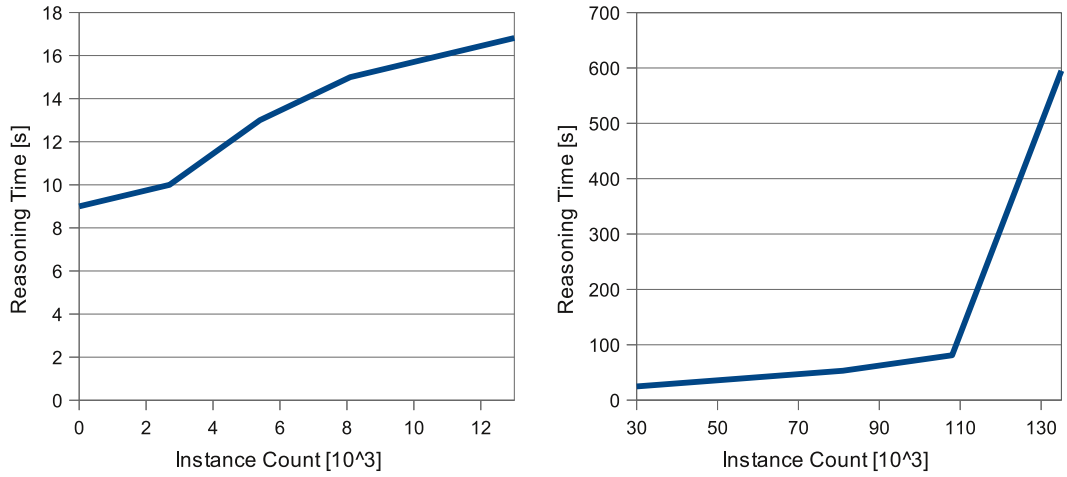
A reasoner is called to evaluate rules on the runtime ontology. For the example from section 4.5, after the reasoning takes place, the reasoner can be queried for instances that belong to the `LargeOldFile` class to receive the results from that rule. After these results have been retrieved, axioms added by the reasoner during rule evaluation are removed from the ontology to reset the state.

A custom module can be installed in the reaction component for a rule or a set of rules, which is configured to watch a certain set of classes (e.g. in the sample case, `LargeOldFile`) and decides how to react when instances of theses classes are found. Reactions can include logging and reporting (e.g. sending SNMP traps to external network managers). More importantly, the reaction module can make use of the CIM interface of the managed system. By invoking CIM methods on the target machine, management decisions can be executed using the existing interface.

5 Implementation and Performance

The translator from CIM to OWL and the runtime system were prototypically implemented in Scala [21], a programming language that integrates object-oriented and functional features and that compiles to Java Byte Code. Java libraries can be used in Scala programs, and OWLAPI 2.2.0 was used for the creation of the ontology. As a reasoner, Pellet 2.2.1 was employed. The CIM client component was custom built.

The conversion of the CIM schema to OWL takes about 35 seconds on an Intel Core 2 Duo with 2 GHz and 2 GB RAM and results in a 12 MB OWL file. Loading the ontology using OWLAPI takes approximately 10 seconds and uses 130 MB RAM on the same computer.



(a) Up to 13,000 instances **(b)** More than 30,000 instances

Figure 4 Reasoning Performance

Figure 4 shows the performance of the runtime system and the time required for reasoning. For this diagram, the computer described above was used and the ontology used for reasoning consisted of two parts: The first part is the CIMOWL ontology with 70,000 axioms, the second part was a varying amount of OWL instances representing `CIM_DataFile`s of a server file system. Each `CIM_DataFile` is represented as a set of 27 OWL instances, which include the instance for the file and one instance for each of the file attributes (file size, modification time, etc.). SWRL rules to categorize files by size (as demonstrated in listing 1) were loaded into the ontology. The test ran on JDK 1.6.0_20. CIM queries were executed on an OpenPegasus ([14]) installation running locally, including a custom built file system instance provider. As the CIM queries take a few milliseconds, their impact on overall performance can be neglected.

In figure 4a the range from 0 to 13,000 instances is shown, which equates to 0 to 500 files. For 2,000 instances and less, the reasoning takes always at least 9 seconds. The graphs show a nearly linear increase in time needed for more instances. Figure 4b shows the limit of the tested system at 130,000 OWL instances, or around 5,000 files. After that, the graph shows a steep rise due to swapping.

The performance results show that the system in its current configuration is not suited for rules that need to be evaluated with a short reaction time (i.e., less than a few seconds). However, a reasonable number of instances for real world applications can be handled. Several changes of the configuration to increase reasoning performances are possible: The reasoning time can be greatly reduced, if only the closure of axioms the rules and instances depend on are loaded into the reasoner. Finding such a closure is not trivial because the corresponding subclass relations and associations have to be taken into account. Another possibility to reduce reasoning time is to reduce the instance count by adding only those file attribute instances relevant to the reasoning result to the ontology.

6 Summary and Future Work

In this paper we presented an approach for management of an IT system, based on the Common Information Model and a heavyweight ontology. Construction of a heavyweight ontology serves multiple purposes: It contains a complete domain model (defined as comprehensively as possible) in a format which is suitable for integration of the managed system with adjacent domains. For example, if processes in an ontology for semantic business process management (see e.g. [10]) refer to physical or logical IT systems, such references can be directly and unambiguously expressed. More importantly, the heavyweight ontology contains model instances that represent the current state of the real world managed system. This allows the definition of behaviour rules that are part of the ontology and reference both model and model instances.

As domain ontology, CIM was chosen, as it is a comprehensive and actively used domain model. As ontology language OWL was applied, as it is the de-facto standard for ontology modelling, and was shown to be the best choice for a translation of CIM. An overview to the approach for the translation of CIM into OWL was given.

A runtime system was presented which is capable of converting the CIM schema into the OWL format and can load the model ontology and statically defined rules in the SWRL format. Moreover, the runtime system acts as a CIM client to gather information about the current state of the managed system and converts this data into corresponding CIM OWL instances. A reasoner that is part of the runtime system is then used to evalute the rules using the model and the current instance data. Reasoning results are used in a reaction

module that can perform arbitrary reactions based each specific rule, in particular issuing CIM requests for calling CIM methods.

By implementing a belief revision operator according to the postulates from belief change theory and adjusted to ontology change, the runtime system component that adds and updates instance data in the ontology makes sure that the ontology stays consistent.

The CIM model is the basis for other standards, such as the storage standard SMI-S (Storage Management Initiative Specification) of the SNIA (Storage Networking Industry Association, [19]), and SMASH (Systems Management Architecture for Server Hardware), a DMTF standard for the unification of the management of data centers. This makes it possible to apply the presented approach directly to ontology-based storage management, where management rules can be specified in SWRL, and ontology-based virtual machine management (for example, VMware server provides a SMASH API). Especially in storage management, usually a large number of rules exist, which can be defined in a coherent manner with the model using the presented approach. As all necessary parts are implemented, application to storage and virtual machine management is investigated in the next step. This investigation will lead to a broader and more detailed evaluation scenario.

Future work also includes the performance optimizations discussed in section 5, especially automatically finding a closure of axioms required for rule evaluation and thus reducing the total OWL instance count. In the long term, the application of the approach to another domain is investigated, Ambient Assisted Living (AAL), where devices and services are combined to provide support for daily life of assisted persons.

References

1 Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50:510 – 530, 1985.

2 Carlos E. Alchourrón and David Makinson. On the logic of theory change: Safe contraction. *Studia Logica*, 44:405–422, 1985.

3 Jorge E. López De Vergara, Antonio Guerrero, Víctor A. Villagrá, and Julio Berrocal. Ontology-Based Network Management: Study Cases and Lessons Learned. *Journal of Network and Systems Management*, 2009.

4 Jorge E. López De Vergara, Víctor A. Villagrá, Juan I. Asensio, and Julio Berrocal. Ontologies: Giving Semantics to Network Management Models. *IEEE Network*, 17(May/June), 2003.

5 Jorge E. López De Vergara, Víctor A. Villagrá, and Julio Berrocal. Applying the Web ontology language to management information definitions. *IEEE Communications Magazine*, 42(7):68–74, July 2004.

6 Distributed Management Task Force. Common Information Model (CIM). `http://www.dmtf.org/standards/cim/`.

7 Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. AGM Postulates in Arbitrary Logics: Initial Results and Applications, 2004.

8 Antonio Guerrero, Víctor A. Villagrá, Jorge E. López De Vergara, and Julio Berrocal. Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL. *Ambient Networks*, 3775:12–23, 2005.

9 Sven Ove Hansson. New operators for theory change. *Theoria*, 55:114–132, 1989.

10 Martin Hepp and Dumitru Roman. An Ontology Framework for Semantic Business Process Management. In *Proceedings of Wirtschaftsinformatik 2007*, Karlsruhe, 2007.

11 Uta Lösch, Sebastian Rudolph, Denny Vrandečić, and Rudi Studer. Tempus fugit - towards an ontology update language. In *6th European Semantic Web Conference (ESWC 09)*, volume 1, pages 278–292. Springer, January 2009.

12 Marta Majewska, Bartosz Kryza, and Jacek Kitowski. *Translation of Common Information Model to Web Ontology Language*, volume 4487 of *Lecture Notes in Computer Science*, pages 414–417. Springer Berlin Heidelberg, Berlin, 2007.

13 Object Management Group. Unified Modeling Language (UML). `http://uml.org/`.

14 OpenGroup OpenPegasus CIM/WBEM Manageability Broker. . `http://www.openpegasus.org/`.

15 Guilin Qi and Fangkai Yang. A survey of revision approaches in description logics. *Lecture Notes In Computer Science; Vol. 5341*, 2008.

16 Stephen Quirolgico, Pedro Assis, Andrea Westerinen, Michael Baskey, and Ellen Stokes. Toward a Formal Common Information Model Ontology, 2004.

17 Márco Moretto Ribeiro and Renata Wassermann. First steps towards revising ontologies. In *Proc. of WONRO'2006*, 2006.

18 Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *ACM SIGMOD Record*, 31(4):12–17, 2002.

19 Storage Networking Industry Association. . `http://www.snia.org/`.

20 Andreas Textor, Jeanne Stynes, and Reinhold Kroeger. Transformation of the Common Information Model to OWL. In Florian Daniel and Federico Michele Facca, editors, *ICWE 2010 Workshops*, volume 6385 of *Lecture Notes in Computer Science*, pages 163–174. Springer Verlag, July 2010.

21 The Scala Programming Language. . `http://www.scala-lang.org/`.

22 Debao Xiao and Hui Xu. An Integration of Ontology-based and Policy-based Network Management for Automation. In *International Conference on Computational Intelligence for Modelling, Control and Automation, 2006*, page 27, 2006.

23 Hui Xu and Debao Xiao. A Common Ontology-based Intelligent Configuration Management Model for IP Network Devices. In *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC)*. IEEE Computer Society, 2006.

24 Hongwei Zhu and Stuart E. Madnick. A Lightweight Ontology Approach to Scalable Interoperability. *MIT Sloan School of Management Working Paper, 4621-06, CISL Working Paper, No. 2006-06*, 2007. http://hdl.handle.net/1721.1/37307.