University of Zurich
Department of Informatics

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

RDF-based IT Configuration Management Database

Daniel Meier
Zurich, Switzerland
Student ID: 04-907-796

Supervisor: Guilherme Sperb Machado
Date of Submission: April 15th, 2011

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

These days, a lot of organizations are highly dependent on their IT infrastructure. Many organizational service provisioning processes are depending on reliable and high-quality IT services. Therefore the area of IT Service Management (ITSM) has become a very important disciple. One of the most established ITSM approaches is the IT Infrastructure Library (ITIL) Version 3, a set of best practices for managing IT infrastructures. ITIL V3 proposes the concept of maintaining a Configuration Management System (CMS), which aggregates the data of multiple Configuration Management Databases (CMDB). This concept supports the management of heterogeneous IT environments as well as guaranteeing the integrity of the maintained infrastructure data.

The developed Semantic Configuration Management Database (SeConD) framework provides the common functionalities required by a CMDB for managing IT infrastructures. In contrast to already existing open source solutions, the developed framework is based on a standardized data model; in this case the Common Information Model (CIM). The goal of this thesis is mapping the object-oriented CIM model to a semantic data model. This has been leading to the development of a CMDB framework which leverages Semantic Web technologies.

Based on the resulting CIM ontology, a semantic CMDB framework has been established. The primary focus of the resulting CMDB lies in the area of storing IT-related Configuration Items (CI) provided by Web Based Enterprise Management (WBEM) enabled hosts. Therefore a Triplestore database provides the required storage back-end capabilities. The SeConD framework only implements the required functionalities in the scope of a single CMDB. Further CMS and business logic parts are not incorporated directly since they are out of the scope of this thesis.

The SeConD framework is an enabler for further work in the area of semantic CMDB research. This allows examining the possibilities of using predicate logic for consistency purposes as well as business rule integration. Therefore further work would include the practical application of this framework in a smaller testbed to collect performance indicators for the reasoning process as well as the performance impact of increasing rule complexity.

Zusammenfassung

Heutzutage sind viele Organisationen in hohem Masse von ihrer IT-Infrastruktur abhängig. Viele betriebliche Leistungserbringungsprozesse sind auf verlässliche und qualitativ hochwertige IT-Dienstleistungen angewiesen. Aus diesem Grund wurde der IT Service Managment (ITSM) Prozess immer wichtiger. Einer der etabliertesten ITSM Ansätze ist die IT Infrastructure Library (ITIL) Version 3, welche eine Reihe von Erfolgsmethoden für die Verwaltung von IT-Infrastrukturen beschreibt. ITIL V3 beinhaltet das Konzept eines Konfigurationsmanagement Sytems (CMS), welches die Daten aus mehreren Konfigurationsmanagment Databanken (CMDB) aggregiert. Dieses Konzept unterstützt die Verwaltung von herterogenen IT-Umgebungen, sowie die Gewährleistung der Integrität von den verwalteten Infrastrukturdaten.

Das im Rahmen dieser Arbeit entwickelte Semantic Configuration Management Database (SeConD) Framework bietet die grundlegend benötigten Funktionen einer CMDB zur Verwaltung von IT-Infrastrukturen. Im Gegensatz zu bereits existierenden Open Source Lösungen basiert das entwickelte Framework auf einem standartisierten Datenmodell, in diesem Fall auf dem Common Information Model (CIM). Das Ziel dieser Arbeit besteht unter anderem darin, das objektorientierte CIM Modell auf ein semantisches Datenmodell abzubilden. Dies führte zur Entwicklung eines CMDB Frameworks, welches die Technologien des Semantischen Web nutzt.

Basierend auf der resultierenden CIM Ontologie wurde eine semantische CMDB erstellt. Der primäre Fokus der resultierenden CMDB besteht in der Verwaltung von IT-bezogenen Konfigurationseinträgen (CI), welche von Web Based Enterprise Management (WBEM) fähigen Hostrechnern geliefert werden. Eine Triplestore Datenbank bietet die dafür benötigten Speicherfähigkeiten. Das SeConD Framework implementiert nur die im Rahmen einer einfachen und einzelnen CMDB benötigten Funktionalitäten. Weitere CMS Teile und Geschäftsregeln wurden dabei nicht direkt berücksichtigt, da sie ausserhalb des Rahmens dieser Arbeit liegen.

Das SeConD Framework ist ein “Enabler” für weitere Arbeiten im Bereich der Erforschung semantischer CMDBs. Es ermöglicht weitere Überprüfungen in der Anwendung von Prädikatenlogik für Konsistenz-Zwecke, wie auch der Integration von Geschäftsregeln. Daher könnten weitere Arbeiten die praktische Anwendung dieses Frameworks in einer kleineren Testumegebung beinhalten, so dass Leistungsindikatoren bezüglich des Reasoning-Prozesses und auch die Auswirkungen komplexer Regeln eruiert werden.

Acknowledgments

I would like to thank the following persons for their help in writing this bachelor thesis:

Professor Burkhard Stiller for giving me the opportunity to write this bachelor thesis at the Communication Systems Group at the University of Zurich. My supervisor Guilherme Sperb Machado for his time, efforts and encouragements during the thesis. And last but not least my family and all my friends who supported me.

Contents

Chapter 1

Introduction

Today a lot of organizations are highly dependent on their IT infrastructure. While this fact is evident for organizations with IT-related core business, it also applies for other non-IT organizations. Many organizational service provisioning processes depend on reliable and high-quality IT services. With the huge growth in organizational IT over the last decades, managing such heterogeneous IT systems has become more and more a very complex task. Therefore the area of unified IT Service Management (ITSM) has grown to a very important disciple. One of the most established ITSM approaches is the IT Infrastructure Library (ITIL). The current Version 3, describes a lot of best practices for IT Service Life Cycle management. A central key concept in ITIL V3 is the management of all IT infrastructure specific information through a Configuration Management System (CMS) that consists of several Configuration Management Databases (CMDB). Those concepts support the principle of providing consistent and reliable IT services in an organizational context.

1.1 Motivation

The current CMDB market includes well known closed source solutions like HP Universal CMDB [1] and IBM Tivoli [2]. Those CMDBs are based on the Common Information Model (CIM), which is an open standard developed by the Distributed Management Task Force (DMTF). There are also several open source CMDBs, but they rely on a proprietary and non-standard data model. Those facts lead to the motivation of implementing an open source CMDB based on CIM.

During the last years the area of Semantic Web has attracted multiple researchers. The Semantic Web concepts allow a machine to process data with implicit knowledge and to infer over the data. Those features disclose new prospects for information integration and knowledge management. Such features are also applicable to CMDBs and could improve the existing technologies in the mentioned areas. Leveraging Semantic Web technologies in a CMS/CMDB could have several positive impacts on development and operation of such systems. For example, using a rule engine could simplify the integration of complex business rules into those systems and reduce the operational costs.

1.2 Description of Work

The work performed in this thesis focuses on the design and implementation of a full-fledged CMDB which is based on Semantic Web technologies. This resulted in the Semantic Configuration Management Database (SeConD) framework. The task includes to two essential points:

- Mapping the CIM model into Semantic Web technologies like RDFS (RDF Schema). The final mapping should respect the CIM standard by maintaining the same organization of classes, attributes and relations of the original model.
- Build an open source CMDB that follows the CIM model. The CMDB should provide mechanisms to easily retrieve and infer information about the related IT infrastructure.

1.3 Thesis Outline

Chapter 2 provides an overview over relevant research in the area of mapping the DMTF CIM model to semantic models. The following Chapter 3 establishes the fundamentals required to understand the context of this thesis. After having set the basic context, Chapters 4 to 6 include the discussion of the taken design choices, implementation specifics and finally the evaluation of the developed framework. This leads to the final Chapter 7 which summarizes the undergone work.

Chapter 2

Related Work

This chapter gives an overview of efforts already done regarding implementations that perform the CIM to OWL translation activity. Besides the well-known modeling of IT infrastructure, the CIM modeling standards of the electric power industry are also accounted. The following paragraphs cover the most interesting approaches and solutions which have relevance in the context of this thesis. After a short introduction and context of each described solution/implementation, a discussion of their benefits and limitations is presented.

2.1 CIMTool

During discussions about the Common Information Model (CIM), people usually refer to the DMTF standard for modeling managed IT environments. A less common fact is, that there is also a CIM model used in the electric power industry. This CIM standard is used to integrate IT systems as well as a standard to exchange information about power transmission and distribution. The International Electrotechnical Commission [3] (IEC) has officially adopted this model as a standard. An exemplary result of the standardization process in the electrical power industry is CIMTool [4]. This tool provides facilities to develop, manage and altering the given CIM standard. At the moment, CIMTool is implemented as an open source Eclipse [5] plugin. It can be used as a general purpose tool that has the capabilities to manage CIM-derived models, profiles and schemas [6].

According to [4] the main features include:

- Read and merge CIM and local UML models in XMI format.
- Browse and validate models.
- Generate equivalent OWL ontologies.
- Design and modify profiles.
- Generate XML schemas and OWL ontologies for profiles.

- Validation of instances according to their profiles.

The mentioned features and efforts seem very promising and interesting in the area of CIM to OWL translation. In the DMTF CIM context, multiple sources refer to CIMTool as a possible approach for such mapping purposes [7, 8]. The main reason for this assumption is its ability to manage and integrate the corresponding semantic models. Therefore, CIMTool appears to be a very interesting solution for other CIM mapping aspects. While this approach seemed very promising back then, nowadays the situation changed. The reason for this is the fact that CIMTool has undergone further improvement and progress with the primary scope on power systems. The latest version (CIMTool 1.8.3) has an explicit scope on power systems, so there is no reasonable approach of using it for DMTF CIM. This implies that the only application area for this tool is modeling and validation as well as designing IEC CIM schemas. Therefore this solution cannot be used for DMTF CIM applications.

2.2 K-Wf Grid OWLTools

The most promising research in the area of CIM to OWL translation has been made in the context of the Knowledge-based Workflow System for Grid Applications [9] (K-Wf Grid) project. This project is a middleware solution to enable the configuration of complex semantic-based grid execution environments. The project includes a *terminated* subproject named OWL Tools [10] that provides different tools in the area of CIM to OWL conversion. Basically three tools were developed:

OWL2OWL Enables the translation of ontology individuals from a source to destination ontology.

CIM2OWL Conversion from DMTF CIM Schema to OWL.

OWL2TEX Automatic production of OWL documentation.

The most interesting component is the CIM2OWL converter. This component provides access to a WBEM Services [11] CIM Object Manager (CIMOM) reference implementation to enumerate and extract the loaded CIM classes. As last step, the CIM2OWL tool converts the extracted CIM Schema to an ontology. While most other approaches try to translate the complete CIM data model from a given format to OWL, the CIM2OWL converter translates the CIM classes that are actually loaded into the CIMOM. The clear advantage of this approach is based on the fact that only the directly affected CIM Core Model, Common Model and Extension Schema classes are translated to OWL. This avoids the huge schema overhead that results in a translation of all existing DMTF CIM classes. Therefore the CIM2OWL converter produces a minimal ontology, but still includes all CIMOM specific Extension Schemas.

An important aspect to mention is, that the CIM to OWL mapping cannot be done in a one-to-one fashion, because some constructs used in CIM are not available in OWL.

To achieve a more accurate mapping, CIM2OWL uses a further meta ontology. After importing the meta ontology, some missing constructs which exist in the CIM vocabulary [7] are made available for OWL. The mentioned meta ontology adds for example Value Maps and default values to OWL.

The features mentioned above made CIM2OWL an interesting approach even if the project is *terminated*. The tool provides valuable foundations, but the distribution includes unresolved problems. For example the last as source code available version (CIM2OWL-1.0.0) uses outdated libraries that include obsolete interfaces. Furthermore the last code base of WBEM Services dates back to November 2004 and was tested against CIM version 2.9. Another problem affects the employed WBEM API based on the Java Specification Request 48 (JSR-48), which has not been finally approved by the Java Community Process. Since CIM2OWL relies heavily on that library, it can be assessed as generic drawback. Additionally, there is no approved WBEM reference implementation for Java available. In fact the CIM2OWL implementation provides a usable CIM Schema mapping, but the code still has implementation flaws in the CIM Instance conversion. There is the possibility that some instance translation issues have been solved in a newer version [12] dated back to November 2008, but it this remains unclear. The newer version is delivered in binary form and the source code is not publicly accessible.

The available CIM2OWL distribution provides useful results for CIM Schema conversion and is until now the most accurate solution for this purpose. Further CIM Instance translations are not possible since the code produces errors and results in an empty translation file. Another problem is the compatibly of the included WBEM Services library, which generates sometimes irreproducible errors for other CIMOM implementations.

2.3 xCIM2OWL

Source Forge lists a project named xCIM2OWL [13] that is programmed in Java. The basic functionality of the "CIM-OWL Conversor" tool is an easy application to obtain the OWL representation of the DMTF CIM model. The converter takes CIM in XML format as input and translates it to a valid ontology. The translation incorporates CIM constructs like aggregation, array and association. Compared to the CIM2OWL tool mentioned in Section 2.2, it lacks of including further CIM specific constructs that are initially missing in OWL. For example the Value Maps used in CIM as well as the read/write qualifiers are not incorporated. This leads to a less exact mapping compared to CIM2OWL. It is vaguer to an one-to-one mapping than CIM2OWL. Furthermore the project neither includes any kind of documentation nor any further information about the context and approaches. The tool can be used in circumstances where only a schema translation is required and the resulting mapping inaccuracies are acceptable.

Chapter 3

Fundamentals

The primary focus of this section is to provide a short introduction to the concepts and models that are necessary to comprehend the assumptions and goals of this thesis. It begins with a short outline of the IT Infrastructure Library Version 3 to derive the principles behind a Configuration Management Database and shows the scope of the affected processes. The following sections introduce the DMTF Common Information Model and provide also an high-level introduction to the Semantic Web technologies and how the components are coupled together.

3.1 ITIL Version 3

One of the most known IT Service Management approaches is the IT Infrastructure Library (ITIL) Version 3. The following subsections provide a brief overview of ITIL V3 in the context of a Configuration Management Database (CMDB). A lot of information about the different Core Disciples has been left out intentionally, since they are not needed to understand the concept of a CMDB.

3.1.1 A Primer to ITIL V3

Organizations require consistent and high-quality IT services. This applies for IT Service organizations in specific as well as for those which need IT only as business support technology. As consequence, the necessity to cover all aspects of IT Service Management (ITSM) in a unified and comprehensive way emerges. Such a fact involves the controlling methods as well as the strategies required to manage any IT service provided by or in the organization, allowing to provide an adequate service quality. To ensure the expected and desired service quality, the Office of Government Commerce [14] (OGC), which is an independent office of the British Treasury, began to develop a generic framework in the late 1980s. Since then, ITIL has become one of the leading frameworks for ITSM. The intend of ITIL is to support organizations in optimizing and managing their IT services

and practices by focusing on people, processes and resources involved in delivering the IT services.

The actual ITIL version proposes a set of guidelines and best practices that describe how ITSM can be implemented and maintained in an organization. ITIL provides the documentation in form of five books [15], each one covering a different aspect of ITSM. The ITIL Service Life Cycle is structured into five Core Disciples which describe the life cycle of any IT service within the organization. Figure 3.1 shows the ITIL Service Life Cycle. Based on the five Core Disciples, the books provide recommendations and

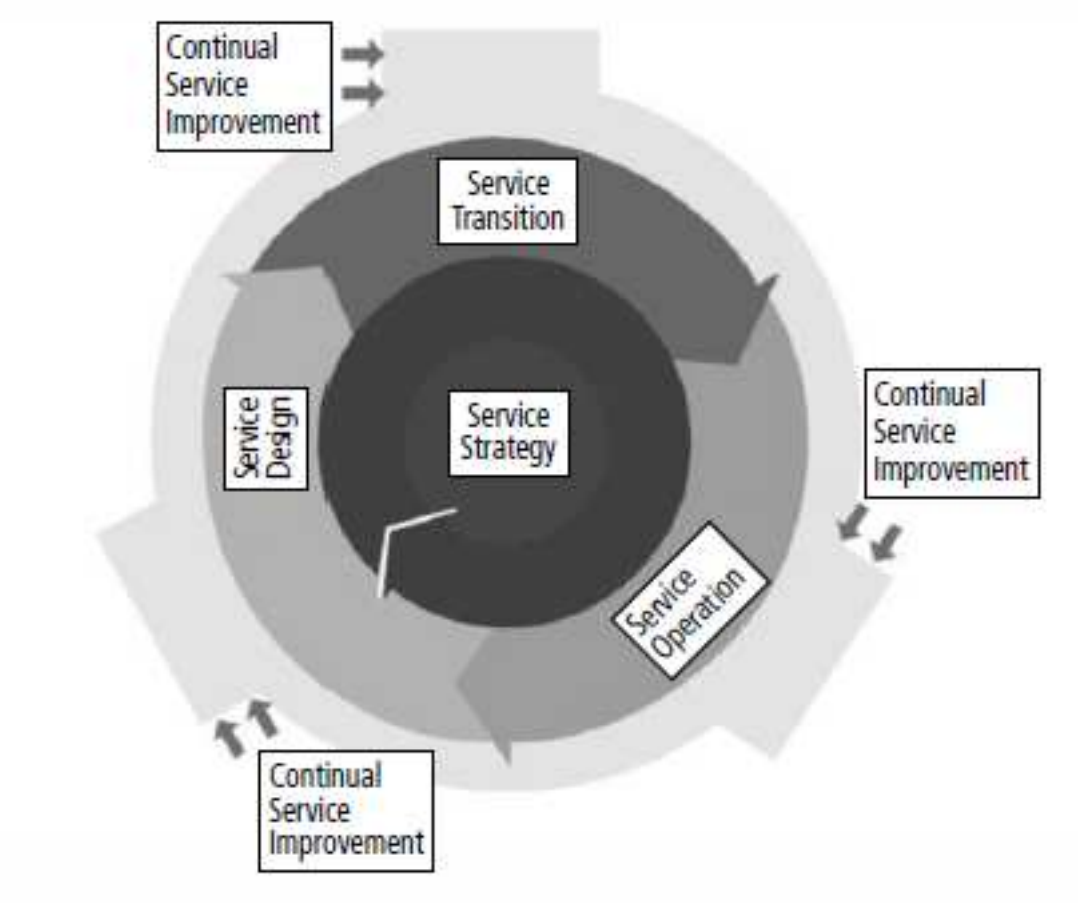

Figure 3.1: ITIL V3 Service Life Cycle [16]

guidance that has to be followed to conduct an integrated approach for ITSM. Adopting those recommendations is required to comply the ISO/IEC 20000 standard specification. The five Core Disciples are:

Service Strategy: This disciple determines the underlying principles for developing policies, objectives, guidelines and processes that are required in the whole Service Life Cycle.

Service Design: Assistance for developing and designing the strategic objectives including an assessment of costs and risks.

Service Transition: Creating a framework which ensures that the designed service is effectively and efficiently implemented in the live environment.

Service Operation: This phase performs all the activities and processes required to run the services effectively.

Continual Service Improvement: This disciple is an overarching phase to improve the effectiveness and efficiency of all the processes involved in all phases of the Service Life Cycle.

The Core Disciple Service Transition includes various principles to support this process. An example is the creation of a standardized framework that supports implementing service and infrastructure changes in a consistent manner. Following basic principles and activities are the key processes of the Service Transition phase [15]:

- Transition Planning and Support
- Change Management
- Service Asset and Configuration Management
- Release and Deployment Management
- Service Validation and Testing
- Evaluation
- Knowledge Management

Those key processes ensure a smooth implementation and transition of IT services. All those processes are partially interconnected and provide further inputs for other Service Life Cycle stages and therefore need to be controlled and closely monitored during the whole life cycle.

3.1.2 Service Asset and Configuration Management

If an organization operates fine and without bigger problems, the business-critical assets are solidly maintained in the normal case. Therefore an important aspect of running an organization is to maintain and protect the business-critical assets as well as maintaining further information about their current status. In ITIL V3 the Service Asset and Configuration Management (SACM) process is an essential part to support that goal. Summarizing it can be stated, that the main purposes of SACM are [15]:

- Identify, control, record, report, audit and verify the Service Assets and Configuration Items in an organization.
- Keep a record of the list of assets and Configuration Items in an organization.

- Report, audit and verify the Service Assets and Configuration Items.
- Account for the Service Assets.
- Protect the integrity of the Configuration Items.

The scope of SACM is not only restricted to IT-related assets; moreover it also includes non-IT assets. This results often in very huge datasets for large and complex IT services and infrastructures. As consequence, the complexity for managing and maintaining the infrastructure raises. ITIL solves this issue by proposing a support system for SACM which is called Configuration Management System (CMS). A CMS holds all the information for about Configuration Items (CI). Those CIs represent the included operating resources, e.g. different components of the IT infrastructure. CIs are saved and maintained in a Configuration Management Database (CMDB). For large infrastructures, ITIL proposes the use of multiple scoped CMBDs which belong to an unified organizational CMS. The CMS is used by all Service Management processes and helps finally to manage the configuration data belonging to the organization. Details about services, CIs and infrastructure stored in the CMDB might need to be updated due to various changes and updates during the Service Life Cycle. The CMS has to guarantee the actuality and consistency of the represented infrastructure and services during the whole time an element exists in the organization. Figure 3.2 shows an example of a complete Configuration Management System. Summarizing, the CMS is a coherent logical model representing the IT infrastructure in the organization.

3.1.3 Configuration Item Requirements

Configuration Items (CI) are primarily hardware and software related items that exist inside the IT infrastructure. They are stored as CI Records in the CMDB. CI Records represent the attributes of a CI and its relations to other CIs. Finally all the CI Records stored in a CMDB demonstrate any managed component in the IT infrastructure that provides required services. The ITIL V3 SACM plan requires a minimum set of specific CI attributes to cover and track the information and changes in an IT infrastructure. The Service Transition book requires the following attributes as a minimal set [15]:

- Unique identifier
- CI type
- Name / Description
- Version
- Location
- Supply Date
- License details

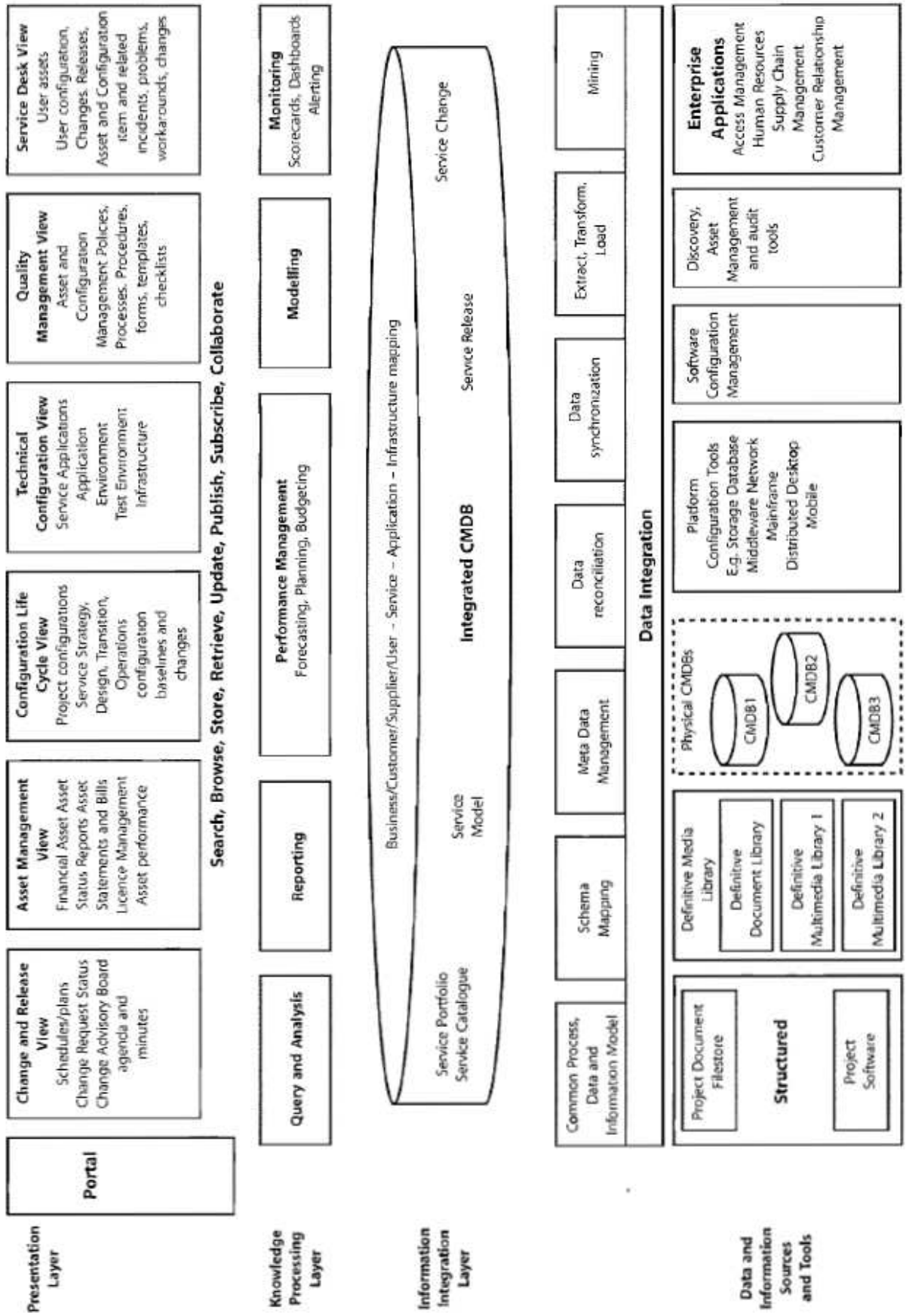

Figure 3.2: Example of a Configuration Management System [15]

- Owner / Custodian
- Status
- Supplier / Source
- Related document masters
- Related software masters
- Historical Data
- Relationship type
- Applicable SLA

3.2 WBEM and CIM

The evolution of the IT industry during the last decades led to huge improvements in efficiency of organizational processes and also generated new business opportunities. On one side those improvements created a higher Return on Investment (ROI) of the IT budgets, but on the other side resulted in more overhead and complexity for managing the IT infrastructure. Managing a complex IT infrastructure leads often to higher Total Cost of Ownership (TCO), which has a negative impact on the ROI. To address that issue, many companies require molded and strong standards to manage their current IT assets and those standards also have to account future growth. Those reasons have led to the foundation of the Distributed Management Task Force [17] (DMTF), which is a standards-based organization with the goal to develop, adopt and unify open management standards for IT environments. The probably most known initiatives are Web Based Enterprise Management [18] (WBEM) and the Common Information Model [19] (CIM). Those two initiatives have the goal to reduce the time and efforts required to manage interoperable IT systems and devices, which leads to TCO reduction and improved time to market. One key aspect is to reduce the development costs by (re)using existing standard models. The following chapters show how CIM is integrated into WBEM and also the concepts behind CIM according to the DMTF specifications and the "CIM & MOF Tutorial" [20].

3.2.1 WBEM and CIM Integration

CIM is an object-oriented management information model which is based on UML. Basically CIM provides a framework to describe the management information. The two foundations of CIM are the *CIM Infrastructure Specification* and the *CIM Schema*. While the infrastructure specification defines the meta schema, syntax, rule and the Managed Object Format (MOF); the schema provides a conceptual framework to describe the managed environment.

Web Based Enterprise Management (WBEM) comprises a set of IT and management technologies that unify IT Management in distributed environments. WBEM itself bases

on the CIM data model. While CIM represents the data model, the other components included in the WBEM stack provide the ability to exchange CIM information in an efficient manner. This includes mappings, protocols and discovery as well as a query language. Figure 3.3 shows the DMTF technology diagram and how WBEM and CIM are related.

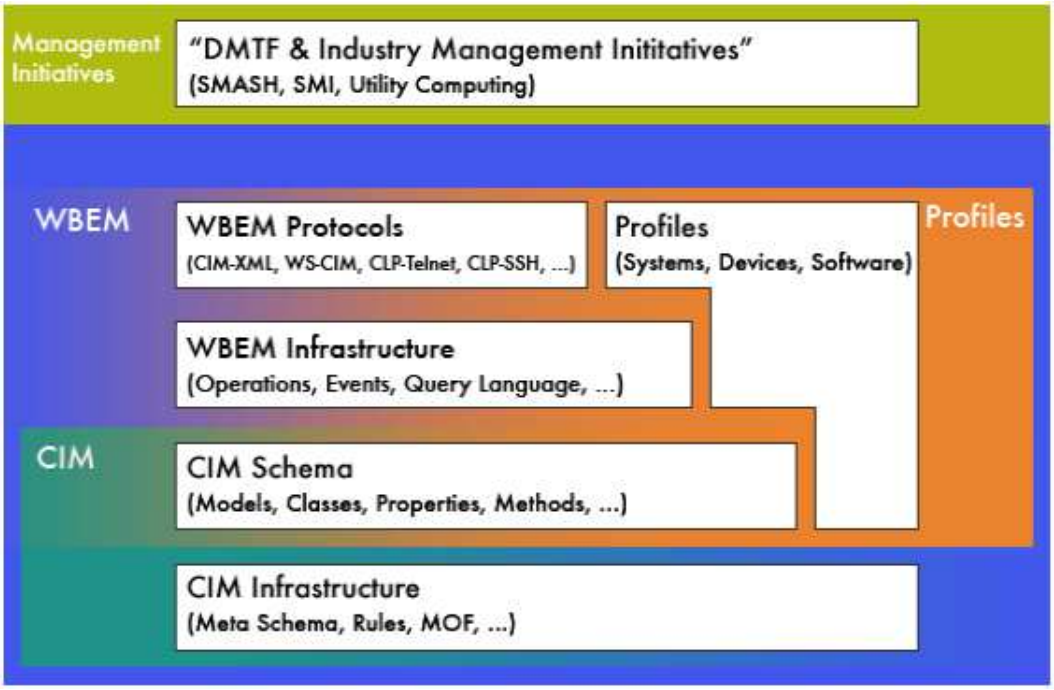

Figure 3.3: DMTF Technology Diagram [17]

3.2.2 The Common Information Model

CIM is a conceptual information model that describes IT infrastructure elements by providing a consistent definition and structure of management information. This description is achieved by using object-oriented techniques to express classes, properties, methods and associations. CIM is in its outline a hierarchical management information model whose elements are represented in the Managed Object Format (MOF). The CIM information model facilitates various interdependencies and relationships between the managed objects. This provides a conceptual view of the managed environment that unifies and extends existing instrumentation and management standards like SNMP and DMI.

As mentioned before, CIM is based on the *CIM Specification* and the *CIM Schema*. The specification defines the details on integrating CIM with other management models. The schema on the other side is responsible for capturing the notions that are used in common areas of management. The notions are implementation-independent.

CIM Specification

The CIM Specification has two fundamental pillars. The first pillar is the description of an object-oriented meta model based on UML. This is the CIM Meta Schema, which describes a formal definition of the model to express itself, its usage and semantics. The Meta Schema is used to express the common elements that are required to clearly represent management applications. With respect to the UML representation, this describes the classes, properties, methods, indications and associations of the model in CIM. Figure 3.4 shows the CIM Meta Schema for a *Named Element*.

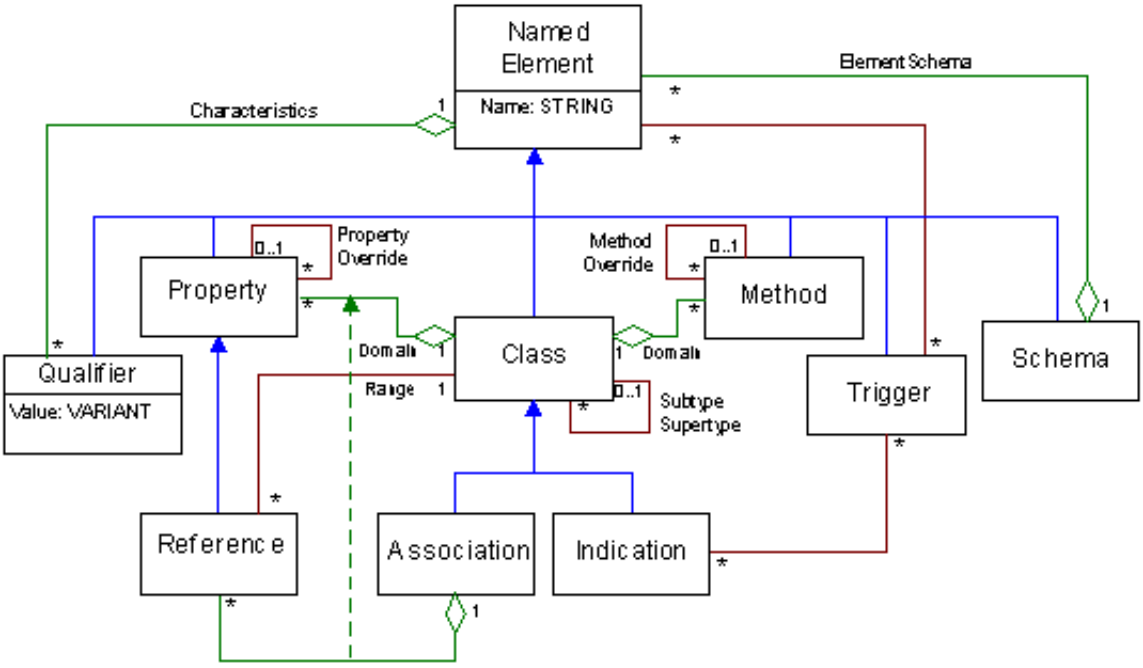

Figure 3.4: CIM Meta Schema for a Named Element [17]

The second pillar of the CIM Specification is the manner of representing CIM management information for information exchange. Syntax and rule specifications of the model are represented in the Managed Object Format, which is based on the Interface Definition Language (IDL). The MOF format defines the meta schema including each meta schema element and the applied rules. Listing 3.1 shows an example MOF class description for a Linux_UnixProcess which is an instance of CIM_UnixProcess. The grammar for the MOF syntax is described in a Backus Naur Form (BNF).

CIM Schema

Management schemas are considered the building blocks for management platforms and applications. This includes device configuration, performance management as well as change management. In the scope of CIM, a managed environment is structured as a collection of interrelated systems which contain a number of discrete elements. The CIM

```
[ Provider("cmpi:cmpiOSBase_UnixProcessProvider") ]
class Linux_UnixProcess : CIM_UnixProcess
{
  uint32 MaxNumberOfChildProcesses ;
  uint32 MaxNumberOfOpenFiles ;
  uint32 MaxRealStack ;

  uint8 terminate();
};
```

Listing 3.1: Example MOF Description for Linux_UnixProcess

Schema provides a well-defined set of classes, associations and methods to represent the managed environment. This includes the *Core Model* as well as the *Common Models*. *Extension Schemas* are tightly coupled, but do not belong directly to the CIM Schema itself.

The foundation of those building blocks is based on the Core Model. It captures the notions applicable to all areas of management as a set of classes, associations, properties and methods. According to [19], the Core Model is broken down into the following sections:

- Qualifiers
- Core Elements/Base Classes (e.g, ManagedElement, LogicalElement, System, Service, Dependency, Component, LogicalIdentity)
- PhysicalElements & Location
- SoftwareIdentity
- Devices
- StorageExtents (subclass of LogicalDevice)
- Collections
- Product and Field Replaceable Units (FRU)
- Statistics
- Capabilities
- Settings
- Power Management

Figure 3.5 shows the top of the CIM building hierarchy. The CIM element on top of the hierarchy is called *ManagedElement*. A ManagedElement acts as a reference for all other class associations in the hierarchy. The subclass *ManagedSystemElement* can represent systems or a component of systems. Furthermore it also can represent any kind of services,

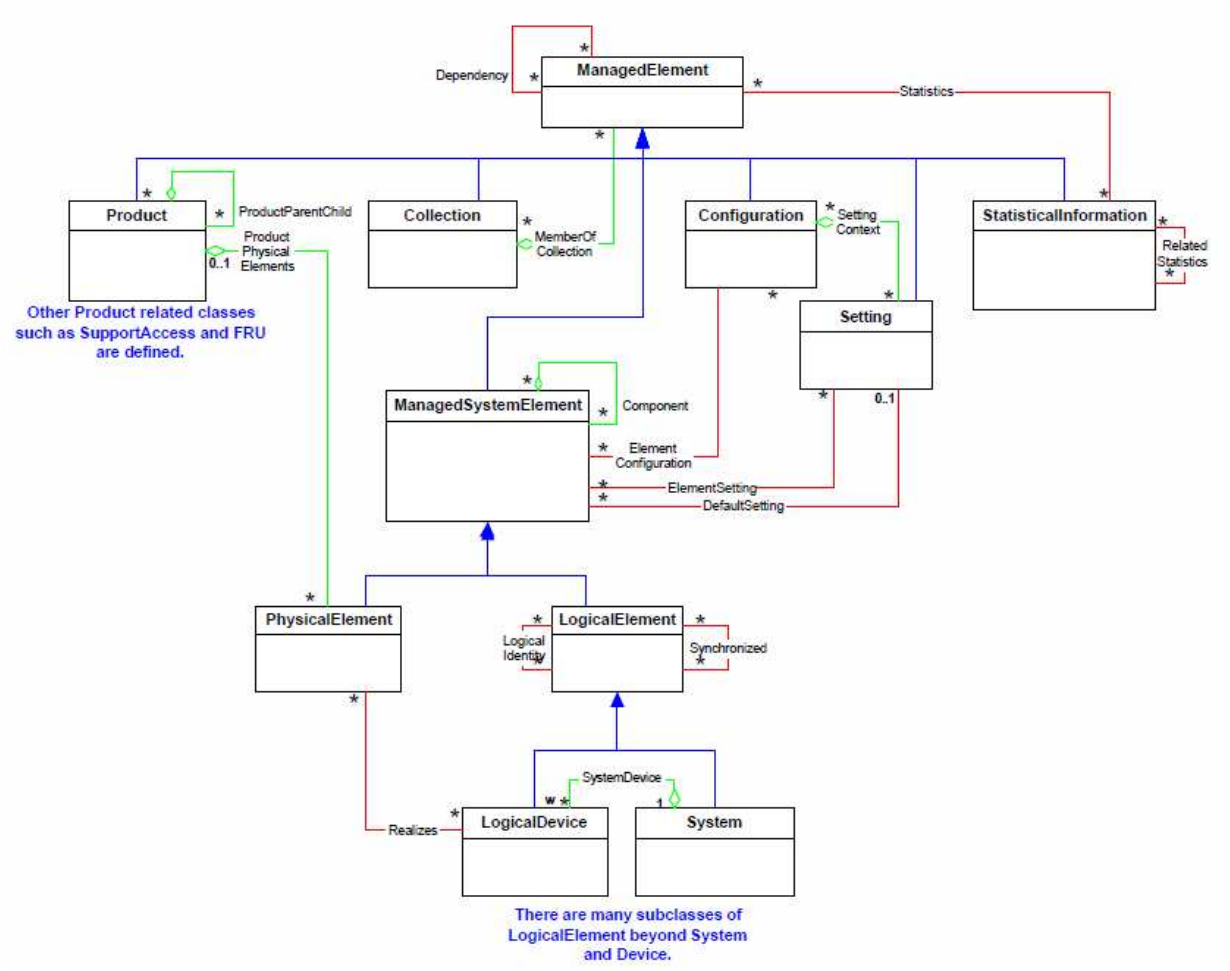

Figure 3.5: Top of the CIM Building Hierarchy [21]

software or networks. The subclasses *PhysicalElement* and *LogicalElement* can represent further definition and specification based on Core and Common Models.

The Common Models represent particular areas of management which are commonly used. Those notions are platform-, technology- and implementation-independent and can include systems, applications, networks and devices. This results in a more detailed view of the managed system. The Common Models for CIM Schema 2.28.0 are:

- Application
- Database
- Device
- Event
- Interop
- IPsecPolicy
- Metrics
- Network
- Physical
- Policy
- Support
- System
- User

Those Common Models can be extended through Extension Schemas, which allow adding platform- or implementation-specific information to concrete classes.

Extension Schema

While the CIM Schema with Core and Common model represents a generic view of management areas, in practice there is usually the need to add implementation- or platform-specific management information. The Extension Schema allows developers to leverage the basic CIM model classes and add their technology specific extensions.

3.3 Semantic Web Components

Semantic Web technologies provide mechanisms that enable machines processing data by understanding the semantics of the provided data. This does not mean that the machine understands the data like a human being, but the machine can classify the processed data through the included semantics respectively metadata. With this basis, further operations on complex knowledge relations are feasible. The term Semantic Web is defined by the formats and technologies that enable it. Those components are based on multiple W3C recommendations like the Resource Description Framework [22] (RDF), Web Ontology Language [23] (OWL), SPARQL Protocol and RDF Query Language [24] (SPARQL) and the Semantic Web Rule Language [25] (SWRL).

The following paragraph assumes prior basic knowledge in the area of Semantic Web technologies and focuses on the interaction between the different components. Figure 3.6 illustrates the major components. Such a high-level overview explains how the different components are coupled together.

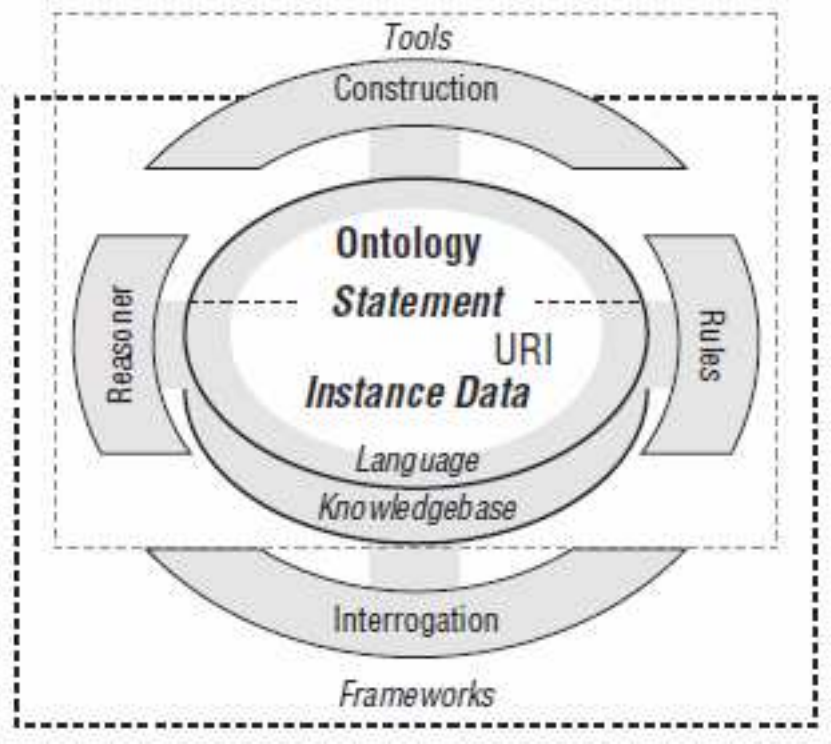

Figure 3.6: Major Semantic Web Components [26]

This allows a description of the most important components [26] in the context of this thesis:

Statement: A statement is a triple that consists of a subject, a predicate and an object. Those triples build the foundation of the Semantic Web.

Ontology: An ontology contains statements about an information domain model. Those statements define the concepts, relations and constraints used in the ontology.

Instance Data: While the ontology defines a generic data concept, the instance data represents information about specific instances.

Reasoner: Reasoners add inference support over semantic data. This provides logical additions for further classification and realization.

Rules: Rule engines provide inference beyond the concepts that can be deducted from description logic.

A Semantic Web framework (e.g. Jena) couples the components together and provides facilities for the fundamental components. Those components are storage, access and inference over the semantic data. Figure 3.7 shows those components in the context of a Semantic Web framework. Based on a valid CIM ontology a Semantic Web framework can be used to implement a majority of required CMDB functionalities in the area of storage and access.

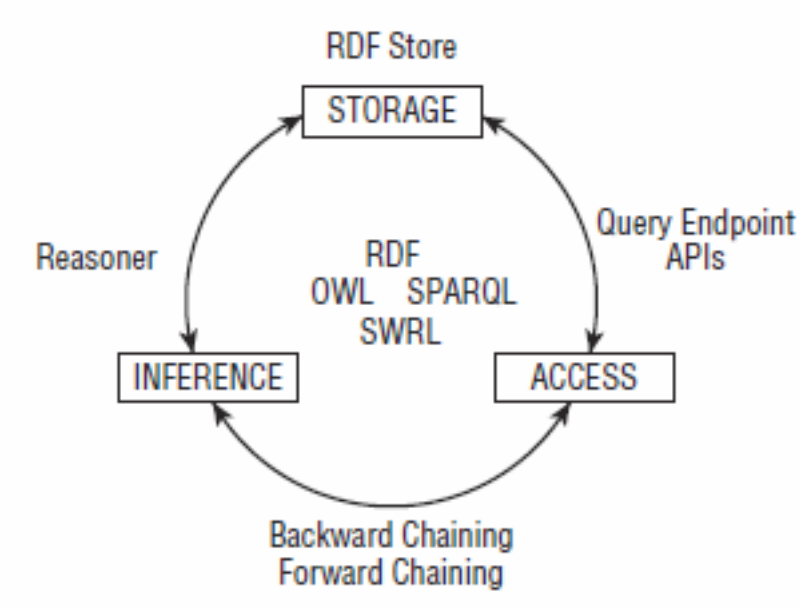

Figure 3.7: Components of a Semantic Web Framework [26]

Chapter 4

Design

Shortly described, this chapter explains how information and configuration data of an IT environment is saved into a semantic CMDB. The following high-level description introduces the generic conceptual steps, while the required tasks to provide such capabilities are more sophisticated. This begins with the utilized Network Management model and continues with multiple follow-up design choices. There is also more in-depth discussion regarding the motivation of using Semantic Web technologies. Furthermore multiple optimization and extendibility aspects need to be taken into consideration. The following sections start with a short excursion to Network Management models and explain why WBEM with its information model CIM has been chosen. After establishing the used data model, the generic application of the required CMDB functionalities is described. This is done by splitting the required functionalities and features into sub-tasks. The description goes hand in hand with the chosen frameworks.

4.1 Network Management Models

At the current state, there are multiple Network Management models with the capability to manage IT environments. Just to mention some examples, there are SNMP, CMIP, DMI and WBEM. Traditionally the area of Network Management was a domain of multiple vendor specific solutions, which resulted in a technology lock-in. Furthermore, some models are incompatible to others or have their own application field. Due to the fact that IT environments are usually heterogeneous, organizations are using different models in the same environment. To assure interoperability and management of such environments, a lot of additional efforts are required. One approach to solve this problem is the concept of mapping different incompatible models to each other. This resulted in several mapping approaches like Domain, Technique and Recast Mapping [27]. The generated mappings suffered loss of information due to the incompatibilities of the incorporated semantics in many cases.

Based on the standardization processes in the past, some models disappeared and other ones remained active. Some good examples to be mentioned, due to the fact that they are

widely known in the area, are the management models SNMP and CMIP. Those between each other incompatible models were the most often used standards in the past. During the 90', other integrated Network Management models appeared. They are based on other technologies than SNMP or CMIP for resource management, even though they are able to incorporate them (at least partially). An example for an integrated Network Management model is WBEM. Many vendors like Microsoft, IBM, Hewlett Packard, Oracle (Sun Microsystems), Novell and Red Hat implemented or adapted parts of the WBEM stack for management purposes into their products.

In the context of this thesis, the usage of the WBEM stack including the CIM information model is a promising approach. This implies using CIM as the data model for persisting information about the IT environment in a CMDB. Since the enterprise CMDB solutions of IBM and HP are also based on CIM, this approach seems reasonable.

4.2 Incorporating Semantic Web Technologies

The majority of CMDB solutions use an information model to represent the IT environment and store this information in a Relational Database Management System (RDBMS). Enterprise level solutions are usually based on CIM as data model and expect enterprise level RBDMS for storage purposes. This approach is absolutely valid and also well established. But with more entities to manage, the complexity of developing, running and maintaining such solutions rises very fast. This leads to possible areas of improvement. Noteworthy is the methodology of storing the information models into the database as well as query specific information from the database. A further aspect is the organizational instrumentation of CMDBs. It can lead to very complex business logic implementations to guarantee consistency, validate information or apply Service Level Agreements (SLA). This is the area where Semantic Web technologies could be leveraged to reduce the complexity of the whole CMDB architecture.

4.2.1 Information Storage and Retrieval

The first complexity barrier starts with information storage. Since CIM is an object-oriented model, it requires mapping to a relational model to persist information in a RDBMS. If no Object-Relational Mapping (ORM) techniques are applied, the mapping task usually results in an extreme huge and complex database schema. Especially the normalization and denormalization processes during database design can bloat the schema. Understanding, maintaining and extending the database leads to a very complex and time-consuming task. Furthermore, querying such a database is very complex without explicit database design knowledge. Even if the knowledge is given, the SQL statements grow very large and complex. They often include a lot of performance consuming join-operations [28]. Such drawbacks can be partially reduced by applying Object-Relational Mapping (ORM) frameworks like Hibernate [29]. While the ORM approach can reduce the mentioned impacts, there are drawbacks in other areas. In practice such solutions

generate an overhead that results in performance impacts. Especially the additionally required join-operations are performance decreasing.

Semantic Web technologies could provide a viable solution to address those issues. The essential approach is to map or translate the CIM model to a semantic model. Such an approach contains issues in the area of mapping accuracy as well as on the performance side during the translation. Depending on the managed IT environment and the performance requirements, those issues can vary on impact severity. This leads to the question which benefits could be gained by using Semantic Web technologies in the area of information storage and retrieval. In the CMDB scenario, data import performance [30] and the required hard disk storage is enough for huge datasets and not a primary concern. The interesting aspect is information retrieval from a Triplestore. While RDBMS use SQL, the analog query language for Semantic Web is SPARQL. In comparison to huge SQL-statements for CMDBs with relational back-ends, SPARQL provides more distinct and less complicated queries. If multiple join-operations are required, SPARQL uses a more compact and understandable notation. This leads often to less development time, because it is easier to formulate the query right the first time and also simplifies the debugging time needed [31, 28]. Another aspect to consider is the SPARQL query-performance, which is fast enough even for huge datasets [32]. Even if the statements related to loading and query performance cannot be directly compared to an RDBMS, the performance is more than sufficient for most CMDB scenarios. This conclusion is also supported by the fact that Semantic Web technologies are designed to handle huge datasets. Besides the fact that SPARQL queries are easier to design, there is another major benefit by using Semantic Web technologies. Any designed or translated ontology can be extended hassle-free with new data model entities. Extending a RDBMS model can become a very complex task, depending on the previous design choices.

Summarizing the comparison described above, leveraging Semantic Web technologies in a CMDB can simplify querying and extending the stored information. RDBMS solutions are more complicated and error-prone in the area of interacting with stored data and extending the database schema.

4.2.2 Business Operations

According to ITIL V3, the data in the CMDB needs to be consistent and valid at any time. Furthermore it is also proposed to integrate also non-IT related information. This means for example to incorporate operational requirements or SLAs in addition. Integrating such features to a CMDB often results in a very complex (programming) task since there is an increase of complexity. In such cases, a semantic integration can be supported with Semantic Web technologies. Using a Reasoner adds inference support to the Semantic Web, which provides a richer set of mechanisms to work with. This feature offers logical additions which provide classification and realization capabilities. Furthermore those features are beyond the basics that can be deducted from description logic and provide support for logic integration tasks.

The mentioned features of Semantic Web technologies support overcoming complexity issues in CMDBs. Complexities based on data validation, integrity assurance and further

operational requirements can be reduced. Especially the ability to merge other ontologies and support larger logical tasks can help enforcing the business requirements of a CMS.

4.3 Conceptual Steps

Very simplified, the task is to retrieve related CIM data and store it in a semantic database. To perform such an operation, a CIM to OWL translation is required. The first approaches began with the translation of the CIM Schema distribution to an ontology. The drawback of such a static translation approach is that the Schema distribution only includes CIM Core and CIM Common. This means that the vendor-specific Extension Schemas are not included in the translation. Such an approach allowed validating the translation concepts, but it is not feasible for a real world approach. The lack of incorporating the Extension Schemas resulted in the loss of host specific CIM information. In WBEM the common practice advises adding host-specific Extension Schemas in MOF format to the CIMOM and then loading a minimal required schema. Using only the CIM Schema in place is an even better solution, since there is no nonessential translation performed. The explained facts lead to three fundamental steps to perform the investigated task:

1. Access the CIM Server and extract the loaded CIM Schema and Instances.
2. Translate the CIM Schema and Instances to an ontology.
3. Store the translated CIM Schema and Instances in a semantic database.

Figure 4.1 visualizes a high-level overview of the three required steps to perform. According to ITIL V3, there is also a CMS involved that collects data from multiple CMDBs and performs further administrative tasks as well as visualization. Those functionalities are out of the thesis scope. The following sections explain the design choices and used framework to fulfill the three steps.

4.4 Accessing a CIM Server

While it is possible to translate only the CIM Core and Common schemas, the practical value of such an approach is very limited. The idea of WBEM is to gather information and data about managed entities of the IT environment. Usually the CIM Schema is loaded into the CIMOM of a managed entity and includes the Schema Extensions. Since the CIM Instances represent the current management data, the focus lies on gathering that specific data. A WBEM Client establishes a connection to any WBEM Server and queries the data. Figure 4.2 shows a very detailed explanation of the WBEM layers according to the Solaris WBEM Services architecture. In short, the client and server communicate after authentication through HTTP (port 5988) respectively HTTPS (port 5989). After establishing such a connection, the CIMOM data can be enumerated.

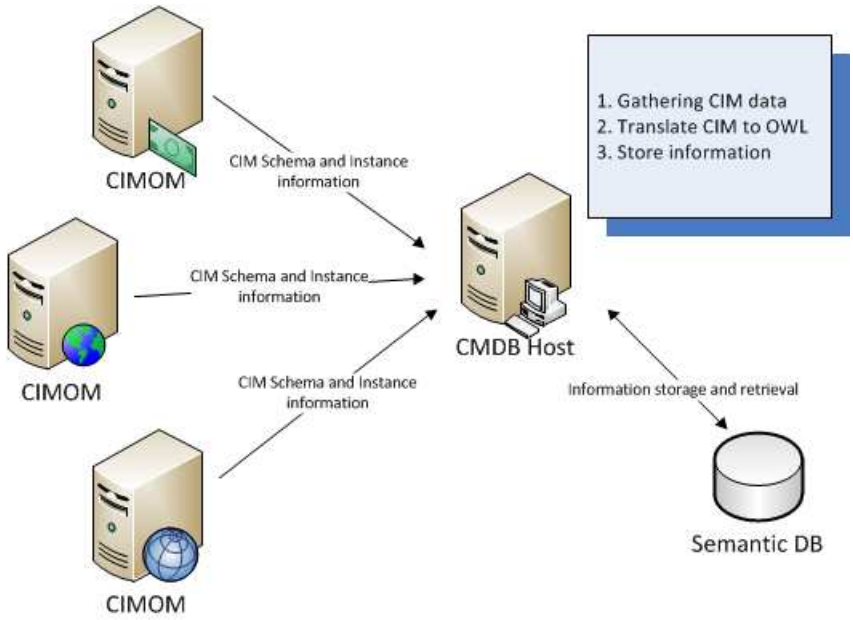

Figure 4.1: Basic Task Description for a Semantic CMDB

4.4.1 Communicating with a CIMOM

The protocol to exchange CIM information is called CIM-XML. It is built on four basic components [20]:

- The Common Information Model (CIM).
- XmlCIM encoding to represent CIM data and operations in XML.
- A set of operations to retrieve and manipulate CIM data.
- HTTP(S) encapsulation.

This implies that CIM-XML uses HTTP(S) as transport protocol, while the payload contains all the CIM operations and responses encoded in xmlCIM. The operations that WBEM defines are described in the "CIM Operations over HTTP" specification [34]. For the purposes in this thesis, the following CIM operations are relevant:

GetClass Retrieves a single CIM Class of the used namespace.

EnumerateClasses Enumerates the subclasses of a specific CIM Class in the used namespace.

EnumerateClassNames Enumerates the subclass names of a specific CIM Class in the used namespace.

GetInstance Retrieves a single CIM Instance of the used namespace.

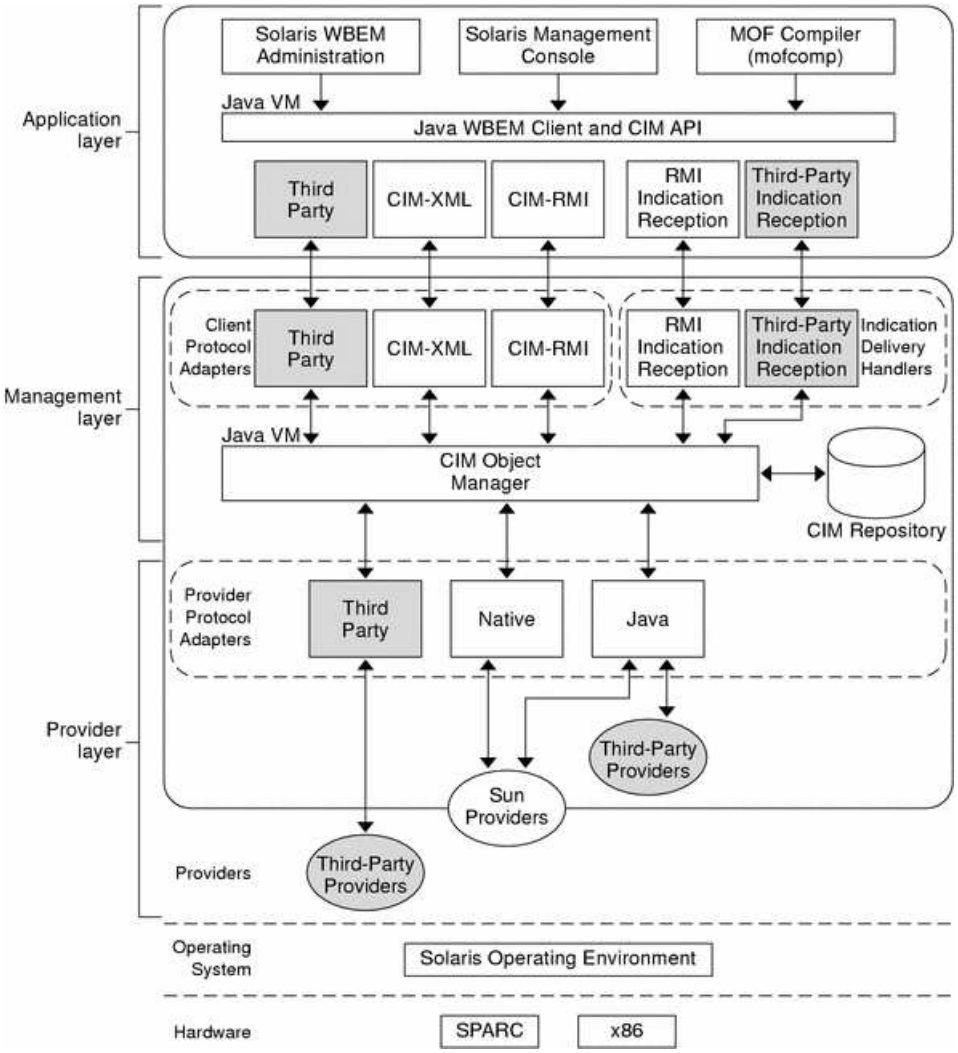

Figure 4.2: Solaris WBEM Services Architecture [33]

EnumerateInstances Enumerates the CIM Instances of a specific CIM Class in the used namespace.

EnumerateInstanceNames Enumerates the CIM Instance names (model paths) of a specific CIM Class.

GetProperty Returns a single property value of a CIM Instance in the used namespace.

Associators Enumerates CIM Classes or Instances that are associated to a specific source CIM Class or Instance.

AssociatorNames Enumerates the names of associated CIM Classes or Instances for a specific source CIM Class or Instance.

References Enumerates the Association objects for a specific CIM Class or Instance.

ReferenceNames Enumerates the names of CIM Classes or Instances for a specific CIM Class or Instance.

GetQualifier Returns a single Qualifier declaration for a specific namespace.

EnumerateQualifiers Enumerates Qualifier declarations for a specific namespace.

Listing 4.1 shows an extract of a *Linux_UnixProcess* instance enumeration in human-readable format, while Listing 4.2 shows the same enumerated extract encoded in xmlCIM.

```
instance of Linux_UnixProcess {
  Caption = "Linux (Unix) Process";
  CreationClassName = "Linux_UnixProcess";
  CreationDate = "20110404145111.000000+060";
  CSCreationClassName = "Linux_ComputerSystem";
  CSName = "octane.localdomain";
  Description = "This class represents instances of currently running...";
  ElementName = "init";
  ...
```

Listing 4.1: Human-readable Extract of an Linux_UnixProcess Instance

```
<INSTANCE CLASSNAME="Linux_UnixProcess">
        ...
        <PROPERTY NAME="CreationDate" TYPE="datetime">
                <VALUE>20110404145111.000000+060</VALUE>
        </PROPERTY>
        ...
        <PROPERTY NAME="Name" TYPE="string">
                <VALUE>init</VALUE>
        </PROPERTY>
        ...
        <PROPERTY NAME="CreationClassName" TYPE="string">
                <VALUE>Linux_UnixProcess</VALUE>
        </PROPERTY>
        <PROPERTY NAME="OSName" TYPE="string">
                <VALUE>octane.localdomain</VALUE>
        </PROPERTY>
        <PROPERTY NAME="OSCreationClassName" TYPE="string">
                <VALUE>Linux_OperatingSystem</VALUE>
        </PROPERTY>
        <PROPERTY NAME="CSName" TYPE="string">
                <VALUE>octane.localdomain</VALUE>
        </PROPERTY>
        <PROPERTY NAME="CSCreationClassName" TYPE="string">
                <VALUE>Linux_ComputerSystem</VALUE>
        </PROPERTY>
        ...
```

Listing 4.2: Extract of an xmlCIM Encoded Linux_UnixProcess Instance

4.4.2 Available Java WBEM Clients

At the moment there are two major Java WBEM Clients available. The first one is the WBEM Services reference implementation and the second one is the SBLIM Java CIM

Client. The common denominator for both implementation is, that they are based on the JSR-48 [35] specification. Besides that, they include different features, tools and partial implementations which are discussed in the following paragraphs.

WBEM Services

The WBEM Services [11] implementation is the reference implementation of WBEM for Java. The origins of this code are based on the WBEM Services for Solaris implementation, which has been adapted and revised. The code underlies the Sun Industry Standards Source License v1.2 and the last release is dated back to November 2004. This implies that the code is designed for a partially outdated CIM Schema. Actually this does not result in essential consequences, since the basic WBEM specifications are still covered and only the CIM Schema has undergone some minor changes and extensions. The distribution includes further WBEM components like:

CIM Object Manager (CIMOM): An example CIMOM implementation is included in the distribution, to support implementation tests.

MOF Compiler: A Java MOF compiler is included to add CIM Schemas to the CIMOM.

MOF-to-JavaBeans Generator: A basic utility to generate JavaBeans based on CIM classes.

CIM Workshop: A GUI tool to browse and alter the CIMOM data.

Preliminary Java API: The implementation is a preliminary snapshot of the JSR-48 specification.

Example Code: Code examples on how to use the API and its features.

SBLIM

Standards Based Linux Instrumentation [36] (SBLIM) is a project that provides an up-to-date implementation of WBEM. The different components of the WBEM stack are optimized for high performance and a minimal memory footprint. The SBLIM stack includes:

WBEM Clients: Besides a command line interface, the SBLIM stack also includes a C and a Java WBEM Client.

CIMOM: SFCB (Small Footprint CIM Broker) is a performance optimized, small disk and memory footprint CIM server.

Providers: Several Common Manageability Programming Interface (CMPI) providers are included, to feed SFCB with management data.

CIM Tools: The Ecute suite includes a modeler, explorer and analyzer to support the development for CIM related software.

The SBLIM Java Client is released under the Eclipse Public License. The older 1.x API is outdated and the current 2.1.x branch is recommended for productive use. A further feature is the inclusion of the Service Location Protocol (SLP) defined in RFC2614 [37]. SLP provides the dynamic discovery of WBEM enabled services in a network.

Conclusion of the Java WBEM Clients

From a technical point of view, the SBLIM implementation is more advanced compared to the WBEM Services implementation. The reason for this conclusion lies in the fact, that the code is more up to date and underwent a lot of development steps which have been tested in productive environments. Since both implementations are based on JSR-48, the SBLIM client seems to be the preferred solution. Both solutions include additional API methods to process CIM objects, which go further than the WBEM specification. The SeConD code is based on the OWL Tools implementation, which uses the WBEM Services library. During the development process multiple API incompatibilities emerged, which finally led to the use of WBEM Services. This choice would be arguable for an implementation in an organization, but since the JSR-48 compatibility is given for both versions and JSR-48 is still in the community process, this fact can be accepted.

4.5 Translation of CIM Data to OWL

This section covers the fundamental aspects of mapping CIM to OWL. Both data models are based on different modeling concepts, therefore mapping CIM specific constructs to OWL is very important. Mapping semantically equivalent constructs is rather simple. CIM-only constructs lead to further issues, because the mapping cannot be done in a one-to-one fashion. Some constructs need to be simplified or can be adapted only partially. As result, it is important that the mapping is unambiguous. The following paragraphs explain the mapping approach and finally also explain which Semantic Web framework fulfills the “semantic level” requirements.

4.5.1 CIM to OWL Mapping

The Semantic Web includes different levels of expressiveness for the described information constructs. This begins with rather simple RDF constructs, advances with RDFS and concludes finally in OWL Full. Each level adds further vocabulary and formal semantic features. To exploit all the strengths of the Semantic Web, the taken approach is to use OWL, because it provides the greatest expressiveness possible in this context. This means that the mapping approaches can utilize the possibilities and features of ontologies like reasoning and predicate logic.

The mapping approach starts with semantically equivalent constructs and then focuses on not fully equivalent constructs. The concepts and approaches used are described in [7, 12]. Semantically equivalent constructs are class definitions and inheritance (to a limited extend) as well as data type attributes and cardinality constraints. The CIM constructs and rules are mapped as exactly as possible. Some constructs and rules had to be mapped in a partial and simplified, but always unambiguous way. Those constructs concerned especially the following areas [7]:

- Data Restriction: e.g. qualifier Value, Value Maps
- Distinction: e.g. qualifier Key, Propagated, Weak
- Redefinition: e.g. qualifier Override
- Access: e.g. qualifier Read, Write
- Versioning: e.g. qualifier Version
- Default Values
- Abstraction: e.g. qualifier Abstract
- Dynamics : e.g. procedures, qualifier IN/OUT

The inclusion of a further "Meta" ontology provides the missing CIM vocabulary. This approach is considered the best mapping solution upon now. Other mapping approaches are possible, but this approach is used in the K-Wf Grid project and proved to be reliable as well as adequate. The obtained mapping for CIM Schema is used as basis for the CIM Instance translation. Table 4.1 shows the final mapping definition for a CIM artifact to its corresponding OWL construct.

4.5.2 Semantic Web Frameworks

The choice of using Java as programming language restricts the amount of considerable Semantic Web frameworks. The three most known frameworks in this area are Jena [38], Sesame [39] and OWL API [40]. Those frameworks however differ in provided functionalities and semantic level. The following paragraphs shortly describe each framework and their features.

Jena is an open source Semantic Web framework developed in the context of the HP Labs Semantic Web Programme [41]. The framework provides an API for all semantic levels like RDF, RDFS and OWL Full. A SPARQL query engine named ARQ is included as well as a rule-based inference engine. Other Reasoners like Pellet [42] are also supported. Besides the typical in-memory storage, Jena also provides multiple storage layers. Jena SDB uses a RDBMS as storage back-end and TDB implements a native Triplestore. Those features support the statement, that Jena is the most advanced and complete framework for developing Semantic Web applications.

Table 4.1: The CIM to OWL Mapping [7]

CIM Artifact	**OWL Construct**
Class	`<owl:Class>`
Generalization	`<rdfs:subClassOf>`
Association (Aggregation, Composition)	`<owl:Class rdf:ID="..."> <rdfs:subClassOf rdf:resource="cim-meta:CIM_Association"/> </owl:Class>`
Property	`<owl:DatatypeProperty>`
REF Property	`<owl:ObjectProperty>`
Method	`<cim-meta:hasMethod>`
Default Value	`<cim-meta:defaultValue>`
Override	`<rdfs:subPropertyOf>`
Key	`<owl:InverseFunctionalProperty>`
Min, Max	`<owl:minCardinality>, <owl:maxCardinality>`
ValueMap, Values	`<cim-meta:CIM_Value>` composed of `<cim-meta:value>` and `<cim-meta:valueMap>`
Deprecated	`<owl:deprecatedClass>` or `<owl:deprecatedProperty>`
Required	`<owl:minCardinality rdf:datatype="&xsd;int"> 1 </owl:minCardinality>`
Experimental	`<cim-meta:Experimental>`
Alias	`<owl:equivalentClass>`, `<owl:equivalentProperty>` or `<owl:sameAs>`
ModelCorrespondence	`<rdfs:seeAlso>`
Read, Write	`<cim-meta:readable>, <cim-meta:writeable>`
Version	`<cim-meta:cimVersion>`
Abstract	`<cim-meta:Abstract>`
Units	`<rdfs:comment>`
Vectors	`<rdfs:comment>`

Sesame is another open source Semantic Web framework which allows working with RDF data. Its origins are based on the On-To-Knowledge project. Sesame supports the SPARQL and the SeRQL query languages. On the storage side, the typical in-memory storage is supported as well as Triplestore and RDBMS back-ends. The major drawback of Sesame is the lack of OWL Full support. There are approaches utilize the lower semantic levels of OWL, but they are insufficient the purpose of this thesis.

The third interesting framework is OWL API, which is an open source framework for working with OWL ontologies. This reference implementation is currently developed by the University of Manchester. The primary focus of this implementation is the creation and manipulation of OWL in an in-memory manner. Therefore further storage layers are not supported out of the box. This matter of fact discourages the use of OWL API for this thesis.

The conclusion of this short assessment in Semantic Web frameworks is, that Sesame and OWL API are not adequate for the usage in this thesis. Jena on the other side provides the full semantic capabilities required as well as including established storage and inference layers. Therefore the use of Jena is undisputed.

4.6 Storing the Translated Data

The best performance for operations on semantic data is delivered by in-memory storage. With huge datasets, the amount of usable memory becomes a limiting factor. To solve this issue, other storage layers are required. In Semantic Web technologies, this is part is provided by RDF stores which are also known as Triplestores. They provide functionalities similar to a RDBMS in the area of storing and querying data. While SPARQL is a query language for RDF stores, the storage model is based on subject, predicate and object triples. Compared to traditional RDBMS, a Triplestore suffers from issues in the area of read and write performance [43, 44]. An approach to reduce this impact is to cluster the Triplestores [45]. Even if those issues could have an impact on the proposed CMDB; there are alternative approaches like portioning the managed IT environment into smaller managed domains. This thesis underlies the assumption that such performance problems can be handled and do not result in unresolvable performance impacts on the storage side.

There are two approaches for RDF stores at the moment. The first approach uses a RDBMS like PostgreSQL as back-end, while the second approach is based on a native Triplestore. Jena provides interfaces for both approaches, SDB for relational back-ends and TDB as a native Triplestore using B+ Trees internally. There are also other Triplestore solutions like Mulgara [46] and OWLIM [47]. Since Jena provides APIs for both storage layers and is used in the CIM to OWL translation, it is obvious to use it for the storage implementation too.

The final storage back-end decision is between TDB and SDB. With loading performance as primary concern, TDB is superior to SDB nearly up to the factor three [48]. In practice the query performance on the datasets is often more important. Statements about the query performance are problematic, since it depends on the underlying ontology structure,

the number of datasets, the query complexity and the required reasoning process [49]. Sometimes TDB is in front and sometimes SDB, or even other RDF store solutions [50]. As a starting point, TDB has been chosen because it is fast enough and a native Triplestore. A drawback of TDB is the fact that Atomicity, Consistency, Isolation and Durability (ACID) transactions are not yet supported. SDB supports acid-style transactions, since it relies on RDBMS back-ends. This shortcoming in the current stage of SeConD is still acceptable. A change to SDB can be done in an uncomplicated manner, since both storage interfaces work in a similar way.

Chapter 5

Implementation

This chapter illustrates how the previous design assumptions and concepts are applied to the SeConD framework. It begins with the initial requirements for the execution environment and continues with a high-level overview based on the package structure. This part introduces the framework components. The chosen package structure is built around two essential functionalities of the implementation. The first one is the CIM to OWL translation and the second one is the storage functionality. Those deeper insights allow discussing the details of the packages based on the interfaces and implementation specific choices.

5.1 Initial Requirements and Package Structure

Before compiling and running SeConD, some initial requirements should be met. Those initial requirements arise from the utilized Java libraries as well as the concepts of Semantic Web technologies. The following initial requirements have to be fulfilled:

JDK 5: Some used libraries require or recommend the usage of JDK5.

HTTP Server: The current SeConD implementation relies on accessing the Meta ontology from `meta-file/Meta` and the translated CIM Schema ontologies through a web server. The required files should be accessible under `http://second-cmdb-host/ontology/cim/`. Therefore the CMDB requires write access to the corresponding file system path where web server documents are stored.

There are several default paths and variables for SeConD that can be customized. This includes the CMDB hostname, the local storage paths and also the CIM to OWL conversion parameters. To customize those settings, the `config.properties` file is provided. Besides those settings, the *Meta* ontology needs to be changed to the actual CMDB hostname (`meta-file/Meta`, line 12 and 13).

The distributed package and directory structure is based on the default values in the properties file. This means that by default the translated ontologies are stored locally under

`Ontologies/` and also under the local path for `http://second-cmdb-host/ontology/cim/`. The storage implementation uses `TDBStorage/` as default path for TDB. An default meta ontology is provided in the directory `meta-file/`. Figure 5.1 shows the dependencies between the packages.

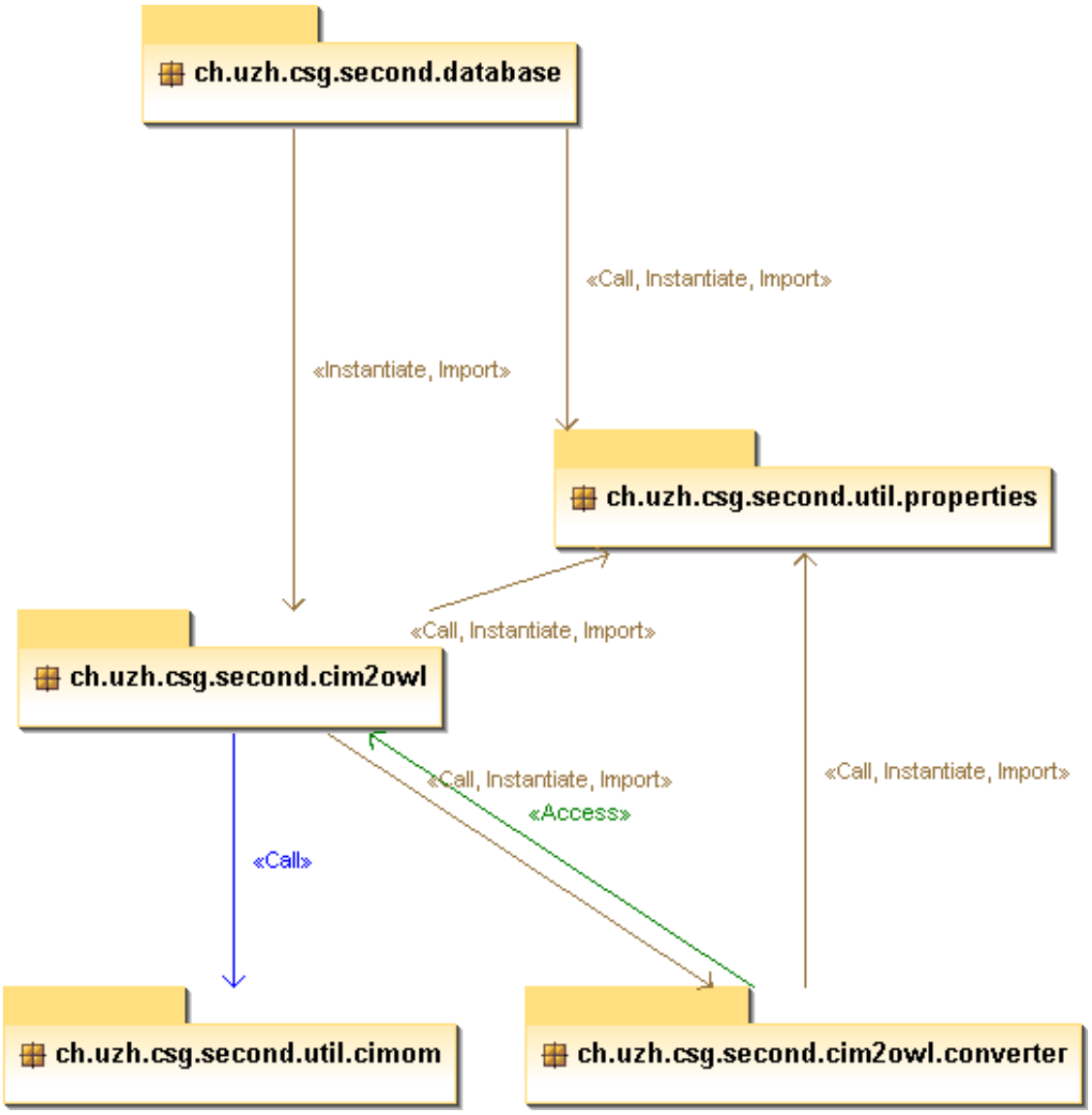

Figure 5.1: SeConD Package Dependencies

The implementation contains three primary packages that can include further sub packages:

ch.uzh.csg.second.cim2owl This package provides the functionalities to translate the CIM Schemas and Instances to an ontology. The utilized mapping structure is hard-coded into the source code and the meta ontology. Translation specific parameters like the inclusion of CIM comments and the usage of default values are loaded from the `config.properties` file.

ch.uzh.csg.second.database The database package represents the primary access point for the SeConD API. It includes the generic *StorageInterface* and a TDB specific *TDBStorageImpl* implementation. The generic interface includes all the API methods to create, read, update and delete the processed CIM Schemas and Instances. The implementation uses TDB as storage back-end and includes further back-end specific methods.

ch.uzh.csg.second.util This package includes sub packages which provide auxiliary functions. Those functions are related to CIMOM connection handling and property file access.

This overview supports the understanding of the following sections which explain the two fundamental parts of the implementation in specific. The first part is the translation implementation and the second one is the storage implementation.

5.2 Implementing the Translation

The translation functionality is always invoked by the storage implementation as soon as new CIM information needs to be added to the database. Figure 5.2 shows the UML class diagram of the cim2owl package. The following description explains how the *CIM2OWL* class is invoked and processes the CIM Schema and Instances:

CIM2OWL The first invocation is based on a non-default constructor which specifies the required translation parameters. The first parameters affect the CIMOM host, user name and password. Those parameters specify the connection to a CIMOM on the default CIM namespace */root/cimv2/*. The last two Boolean parameters specify the designated translations of CIM Schema and Instance. This allows to initializes the *convert()* process, where the desired CIM translation are started. The process incorporates also the properties of the translation and the used ontology storage paths.

CIM2OWLFlatConverter This class represents the implementation of the abstract class *CIM2OWLConverter*, which provides the functions to translate extracted CIM classes and CIM instance paths to the destination ontology. The principle behind this process is a flat translation. This means that CIM is represented in portioned namespaces. This fact requires handling implicit CIM semantics like nested naming and implicit references [51].

The CIM Schema flat translation needs to incorporate the implicit CIM semantics, which is done by the following steps:

1. Acquire all simple CIM classes that are not associations.
2. Generate a simple class subsumption hierarchy.

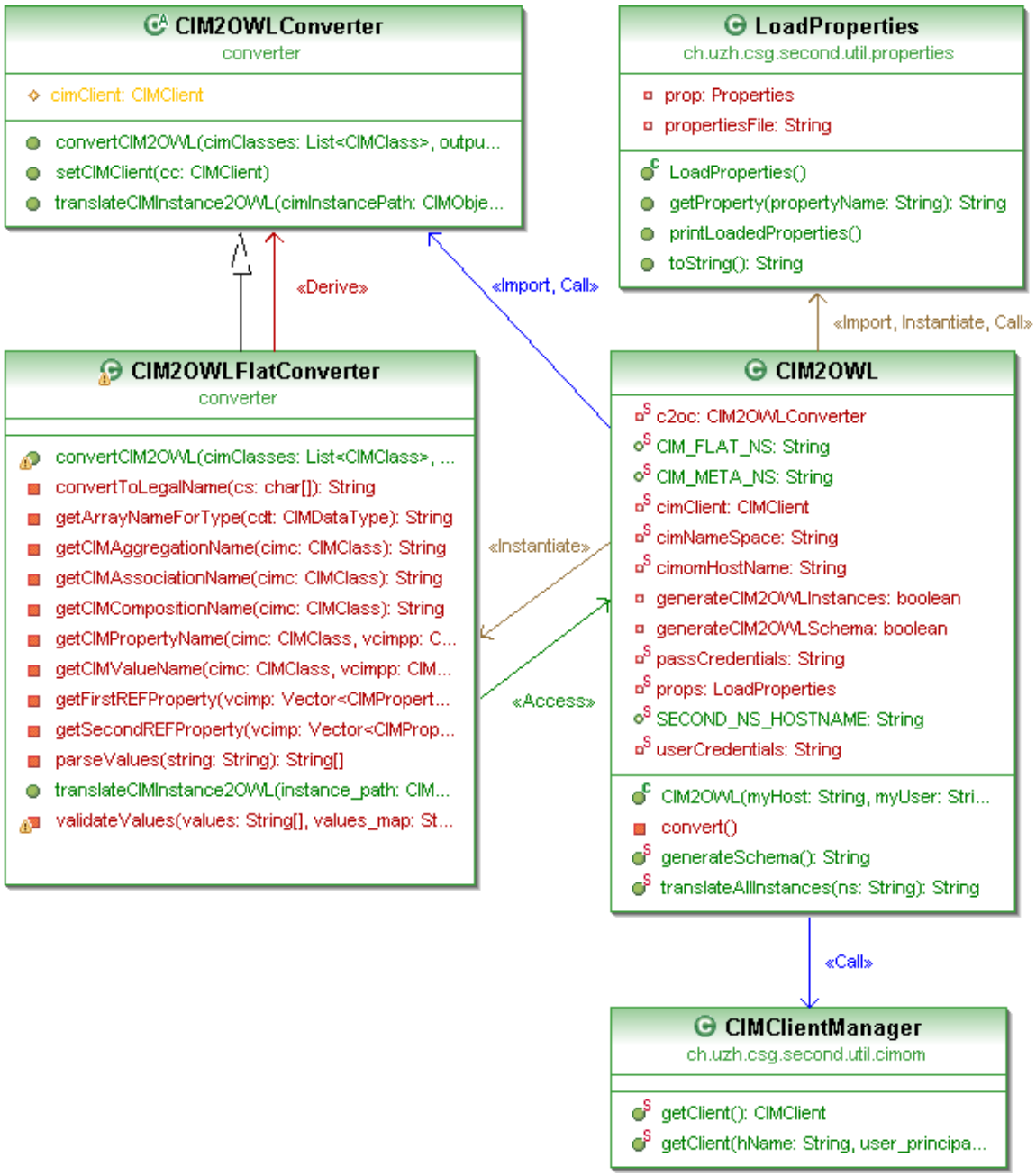

Figure 5.2: *CIM2OWL* UML Class Diagram

3. Generate all associations as object properties. This step includes also the CIM qualifiers Aggregation, Association and Composition.

4. Generate simple object properties.

Those steps generate a valid OWL ontology for the CIM Schema that is used as basis for Instance translation. The Instance translation maps a regular CIM class to an ontology individual and traverses all its properties. Each translated individual includes a timestamp of the current time in milliseconds as suffix. This notation allows establishing a time line for all translated instances. Furthermore the actual CIM object path is added as a *<dc:identifier>* property to include it direct way. Figure 5.3 shows the translation of the CIM hierarchy to the destination OWL2 ontology. The CIM class *Linux_ComputerSystem* holds a translated CIM instance named *Linux_ComputerSystem1302112524119*.

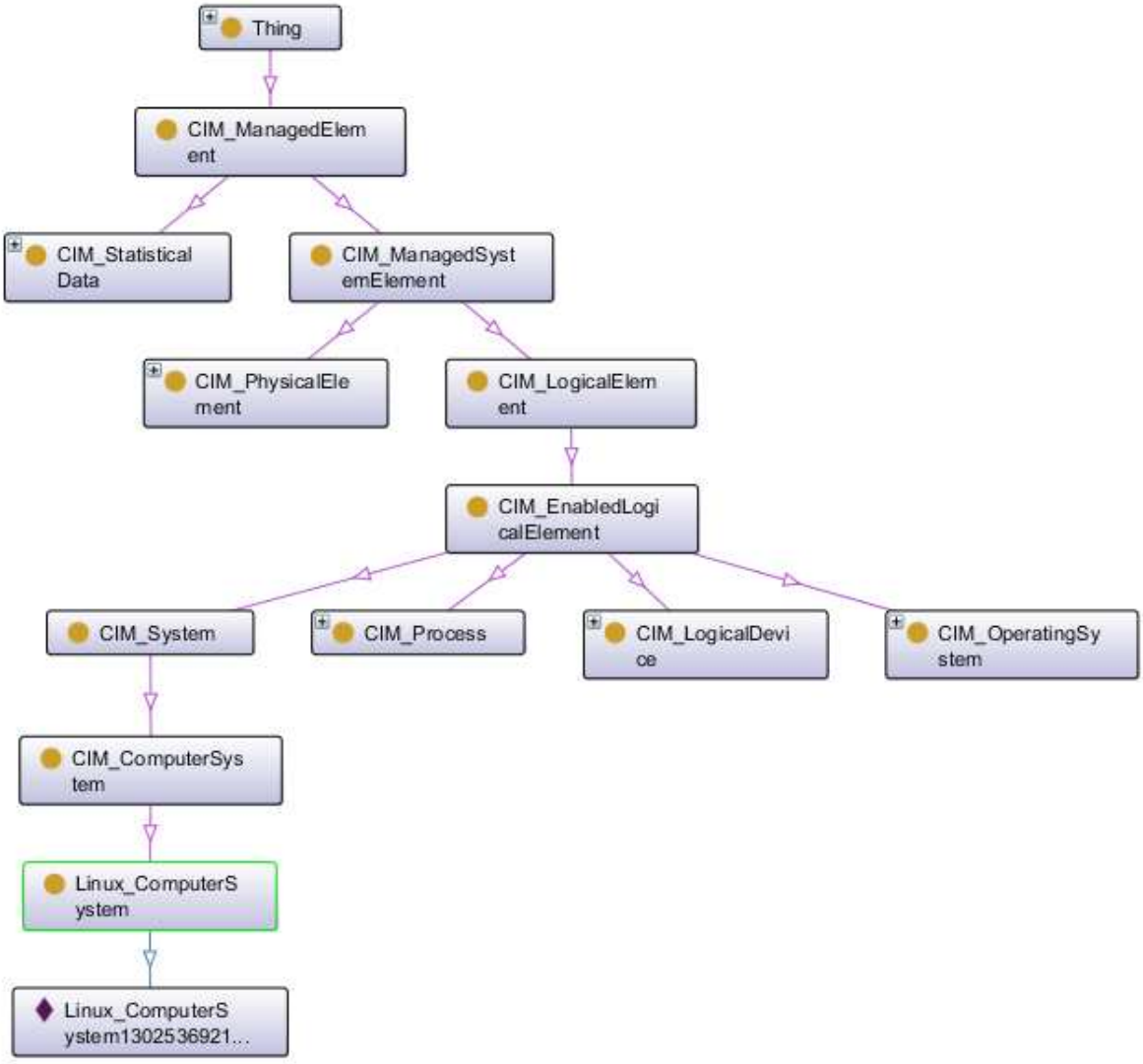

Figure 5.3: CIM Ontology for the Extension Schema Class Linux_ComputerSystem

5.3 Implementing the Storage Layer

The *StorageInterface* class represents an interface for the defined functionalities. Figure 5.4 shows the UML class diagram of the database package. The typical Create, Read, Update and Delete (CRUD) functionalities are provided through this interface. Those functionalities take care of handling semantic data stored in (ontology) models. This includes the synchronization of the used in-memory ontology model with the model used for the storage layer. The API provides a set of useful and required CMDB operations in the context of a unifying CMS. An explicit SPARQL interface was not included do to the fact that Jena already provides such an interface through ARQ. The different *listAllCIMxxx()* methods provide functionalities to retrieve information about the managed hosts. This includes the host addresses that are maintained through the CMDB and also information about all or specific CIM Instances and classes. The returned values are strings and not ontology classes or individuals. At the moment, there is no application for a CMDB to manipulate ontology classes and individuals outside of the direct CMDB context. Furthermore public methods that provide capabilities to manipulate the ontology classes and individuals directly could lead to data corruption and inconsistencies. This restriction is based on the fundamental idea of having a CMS (or an integrated CMDB) that provides further data processing and management capabilities for all managed entities at the Information Integration Layer.

The *TDBStorageImpl* implementation uses TDB as storage back-end. The implementation maintains two semantic models. The first model is an in-memory ontology that is used as "working model" for all operations on the semantic data. After each in-memory operation, the "working model" is explicitly synchronized with the "storage model" used for TDB. The reason for this explicit synchronization lies in the fact that TDB does not support ACID-style transactions. One way to get around this issue would be the usage of SDB as storage back-end. To reduce the impact of this drawback, only successfully translated ontologies are processed at the storage layer. After each update on the "working model", a synchronization with the "storage model" is carried out. This minimizes the drawback of not supporting rollback operations for writing and update purposes. Read-only and SPARQL queries are not affected, but the issue should be considered if the SPARQL Update [52] language is used on the SeConD framework.

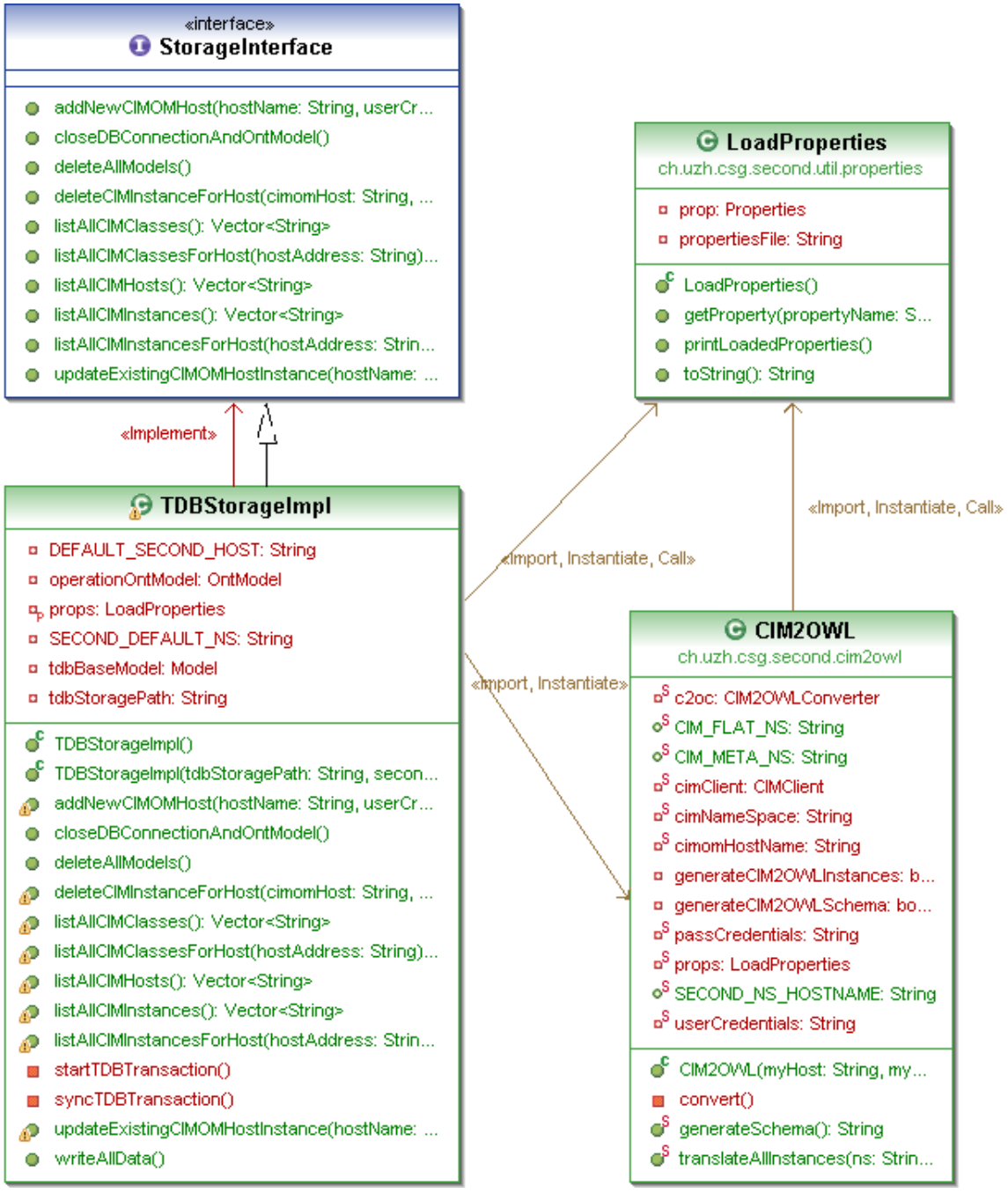

Figure 5.4: *StorageInterface* UML Class Diagram

Chapter 6

Evaluation

A final evaluation of the provided SeConD framework can be done from different points of view. The first part of the evaluation covers a set-actual comparison of the given thesis goals as well as the translation performance. While such a comparison can be done inside clear context boundaries, a qualitative comparison allows discussing the aspects and advantages of the taken approach. Those aspects are based on the fact that the provided solution is the first publicly known attempt in implementing an open source CIM-based semantic CMDB. This leads to the second part of the evaluation chapter, which is represented as a high-level qualitative comparison. This aspect is necessary, because there are a lot of implementation and context specific variables that can lead to very different performance behavior. Therefore the chapter contains a discussion of possible bottlenecks and advantages of a semantic CMDB.

6.1 Thesis Goals and Related Aspects

The thesis goal comprised the design and development of an open source RDF-based CMDB for IT management purposes. The implemented solution should be based on the CIM model. The following sub-goals had to be achieved:

- Implementation of a RDF schema that represents the CIM model (full model or a subset).
- Design and implementation of a scalable CMDB including well-defined interfaces to persist data (respecting the CIM model).
- Design and implementation of an abstract layer (e.g. Java library) that enables the use of the CMDB by external software.

The first goal and foundation of this thesis required a mapping of CIM (Core and Common) to a RDF schema. First partial mapping approaches pointed out that a deeper semantic level than RDF was required to incorporate all CIM semantics. This finally led

to the use of OWL Full, which provides a better and more precise mapping result. The initial mapping focused on the CIM Core and Extension Schemas, which proved to be sufficient for the first mapping stages. The decision to use the WBEM stack for further development led to a solution that includes the Extension Schemas. Furthermore, only the actual used CIM classes and Schema Extension classes are translated. A clear advantage of this approach is the avoidance of unnecessary translations and therefore minimizing the host specific ontology overhead. From this perspective the goal was not only achieved, furthermore the used approach exceeds the initial requirement.

At the current stage, the SeConD framework provides an abstract layer to access the provided functionalities. In regard to further development, a generic storage interface has been designed. The design choice to use an internal in-memory ontology model and a second model for the storage layer allows uncomplicated integration of other storage back-ends.

Last but not least, there are some remarks on the provided distribution. The framework still produces some CIM error messages, but they have no practical impact on the translation results. It is a problem of using an old WBEM reference API on actual CIM. This issue can be addressed as soon as JSR-48 will be accepted and officially integrated into Java.

6.2 Translation Performance and Network Usage

A very important aspect of the overall performance is the translation performance itself. Therefore an exemplary testbed has been designed to gather some reference points on the translation performance. Figure 6.1 shows the testbed design. This rather simple design includes a CIMOM host to provide the management data as well as a CMDB host running SeConD. Both machines are interconnected through Fast Ethernet.

CIMOM host configuration:

- Operating System: Ubuntu 10.10 Server (x64)
- Hardware: Intel Core i5-750, 8GB RAM
- CIMOM: SBLIM sfcbd 1.3.8 (including cmpi-base provider)
- CIM Schema: DMTF CIM 2.25.0

CMDB host configuration:

- Operating System: Windows 7 Professional (x64)
- Hardware: Intel Core 2 Duo,4GB RAM
- JDK: Oracle JDK6 (1.6.0_24)

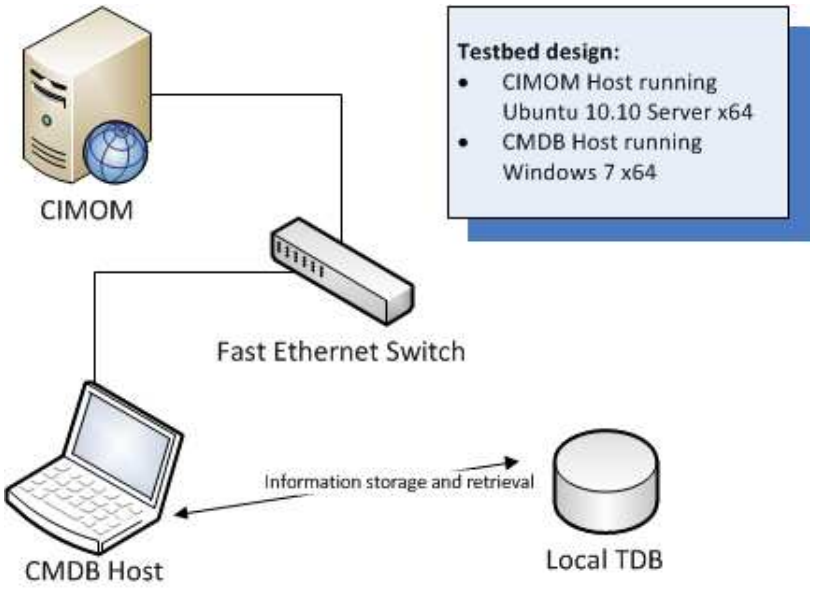

Figure 6.1: Testbed Design

The performance measurements are based on the Java Execution Time Measurement [53] library. To achieve reproducible results, the translation performance has been measured for 10 iterations of a CIM Schema and Instance translation. Furthermore the CIMOM host had to be idle to assure identical measurement conditions. Another important aspect to classify the measured results is the number of actually maintained CIM Schema and Instance classes. Table 6.1 summarizes the resulting measurements. It is evident, that the CIM Instance translation seems to perform rather poorly. The bottleneck occurs while traversing the properties of the resulting ontology individuals. Each affected CIM Schema class and property has to be processed and the corresponding data type values are assigned to the resulting ontology individual. A further approach to trace the origins of the bottleneck was based on the traffic observation between CIMOM and CMDB host. During one complete translation, a continuous dataflow of *totally 27.30 MB* has been monitored. More precisely the CIMOM *transmitted 15.80 MB* and *received 11.50 MB*. Approximately 94% of the measured traffic was related to the Instance translation. Table 6.2 shows the traffic measurements for one complete translation based on the same test conditions as stated before.

The preceding chapters attested that the storage back-end performance itself should be sufficient. The bottleneck occurs during the CIM Instance translation. The bandwidth usage itself is not excessive, but rather processing the exchanged data. An approach to reduce the impact of the required Instance translation time could base on smaller and sliding Instance update batches.

Table 6.1: Translation Performance over 10 Iterations

	# Classes	**Average**	**Min**	**Max**	**Total**
CIM Schema	168	6'306 ms	5'954 ms	6'909 ms	63'065 ms
CIM Instance	93	365'269 ms	321'940 ms	380'149 ms	3'652'693 ms

Table 6.2: Traffic Measurements for a Complete Translation

	TX	**RX**
CIM Schema	1.490 MB	0.0667 MB
CIM Instance	14.31 MB	11.43 MB
Total	15.80 MB	11.50 MB

6.3 High-Level Qualitative Comparison

Actually there is no comparison to other open and closed source CMDBs that use CIM as data model *and* include Semantic Web technologies. The SeConD framework seems to be the first publicly know approach in this direction. Therefore a high-level qualitative evaluation of expected advantages based on the new CMDB approach is especially preferable. Moreover this implementation could act as an enabler for further research in the area. The following paragraph covers the expected improvements in data query and business logic integration.

Almost every well-established CMDB solution utilizes a RDBMS storage back-end. Maintaining and extending optimized database schemas can become a very complex task. On the other side an ontology based schema can be developed, maintained and extended in a less elaborate way. Ontology editors like Protégé [54] are very useful tools for that purpose. While the mentioned aspects have their focus on development and operation, the practical integration and usefulness factor in a CMDB provides more concrete advantages. Creating complex queries on relational database schemas can lead to an error-prone task that under certain conditions involves a lot of debugging time. Compared to SQL, SPARQL queries are easier to create and understand. Even if those features seem to be interesting, the principal value of leveraging Semantic Web technologies lies in an area with direct business impact. The principal area of improvement is based on the business logic for CMS and CMDB systems. Such business logics can contain consistency checks, SLA integration or further operational rules. Those rules are usually included in the database schema or provided by stored procedures as well as in program code. Especially multiple and complex rules can result in huge code fragments that are difficult to maintain and extend. With Semantic Web technologies, the rule engines provide a way to leverage predicate logic. Creating business logic rules based on predicate logic can simplify this process and finally lead to faster time to market. Depending on the requirements of the complete CMS solution, those aspects can reduce the Total Costs of Ownership for maintaining the IT infrastructure of an organization and help improving the business processes.

As promising as the stated concepts sound, there are still many unknown variables that can result in various forms of performance impact. Even if those factors are out of the thesis scope, they are still noteworthy. At the current state, the performance impact of reasoning is not fully investigated. Depending on the number and structure of the stored triples as well as the amount of the incorporated rules, the reasoning process could result in serious performance hits [49]. Those impacts can be reduced by choosing a smaller CMDB domain or changing the update intervals. Another promising approach is a clustered TDB to improve load time and query throughput [45]. This leads to the final conclusion that a trade-off between CMDB size and performance has to be evaluated. Such a trade-off

depends on multiple deployment specific parameters, which cannot be generally evaluated at the current stage of Semantic Web development.

Chapter 7

Summary and Conclusions

This thesis describes the design and implementation of an open source IT Configuration Management Database according to the IT Infrastructure Library Version 3. The resulting SeConD framework makes use of the Common Information Model as data model and leverages Semantic Web technologies. A CMDB implementation that combines both concepts has not yet been seen - at least in public domain.

The first two chapters comprise an introduction into the thesis topic as well as a discussion of thesis related work.

The succeeding chapter three introduces the fundamental components and concepts of a CMDB. This includes an introduction into the relevant ITIL and WBEM concepts. The first part provides an overview of the adapted ITIL and Web Based Enterprise Management concepts. A high-level overview about Semantic Web components concludes this chapter.

Chapter four illustrates the design choices for the CMDB as well as their justification. This includes the motivation for choosing CIM as data model and the expected benefits of leveraging Semantic Web technologies.

This leads to chapter five where the applied design choices result in the implementation of the SeConD framework. The chapter describes the provided framework based on the interfaces and internal operation.

Chapter six finally evaluates the framework in the context of the thesis task and depicts performance measurement points. The chapter is closed up with a high-level qualitative comparison between existing and newly introduced CMDB concepts.

7.1 Conclusions

The SeConD framework is based on a well-established and widely accepted data model. This fact enables the underlying data model to be in tune with the ITIL V3 concepts for a CMDB. Applying Semantic Web technologies promises further overall improvements in

the area of CMDB operation. The expected benefits could result in diminishing Total Cost of Ownership for IT environments. Regardless how promising the expected benefits appear in theory, they are not yet proven in a live environment. The size and diversity of heterogeneous IT environments as well as the business operation requirements can lead to not yet recognized issues. The scalability and possible bottlenecks can be roughly arranged, but only concrete appliance provides reliable indications of realizable benefits. There is also the possibility that Semantic Web technologies are poorly conceived for CMDB specific purposes at the current stage. Furthermore, it is possible that the Semantic Web concepts are overhyped. Nevertheless, the developed framework is an *enabler technology*. It provides a solid basis for further research in this area. Therefore it facilitates an examination of the expected benefits. On one side, this can lead to the conclusion that Semantic Web technologies are not fully applicable for CMDB purposes. On the other side, it could reinforce the research of semantic CMDBs and perhaps lead to new CMDB capabilities. SeConD is considered as an advance towards answering those questions.

7.2 Further Work

Since the SeConD framework is a prototype, there are several areas of improvement. One aspect is the further development of SeConD itself. This begins with a better scoped exception handling and continues with the integration of a logging facility. Especially the poor CIM Instance translation performance should be considered the next working point. Helpful in this context would be the final approval of JSR-48. Since it became quiet around this specification request, a change to the SBLIM WBEM Client could be considered as an alternative. This would allow further source code improvements.

Another aspect is the full integration of a Reasoner respectively a rule engine. This would allow exploring the performance impacts of reasoning and inference. This functionality finally enables further research in a live environment. Therefore it facilitates a review of the expected benefits as well as the investigation of bottlenecks and their disposal. Furthermore common limitations and required trade-offs can be debated.

Bibliography

[1] HP Website. Hp universal cmdb software. Available online at https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-25^1059_4000_100__. Last visited on: April 15th 2011.

[2] IBM Website. Tivoli change and configuration management database. Available online at http://www-01.ibm.com/software/tivoli/products/ccmdb/. Last visited on: April 15th 2011.

[3] IEC Website. Welcome to iec - international electrotechnical commission. Available online at http://www.iec.ch/. Last visited on: March 21st 2011.

[4] CIMTool Website. Welcome to cimtool.org. Available online at http://wiki.cimtool.org/index.html. Last visited on: March 21st 2011.

[5] Eclipse Website. Eclipse - the eclipse foundation open source community website. Available online at http://www.eclipse.org/. Last visited on: March 21st 2011.

[6] Langdale Consultants. Cimtool: Bridge from cim to applications. Available online at http://files.cimtool.org/CIMUGPresentation.pdf. Last visited on: March 21st 2011.

[7] Marta Majewska, Bartosz Kryza, and Jacek Kitowski. *Translation of Common Information Model to Web Ontology Language*, volume 4487, pages 414–417. Springer Berlin Heidelberg, 2007.

[8] *Cracow'06 Grid Workshop, Vol 2: K-Wf Grid: the knowledge-based workflow system for Grid applications: October 15–18, 2006 Cracow, Poland*. Academic Computer Centre CYFRONET AGH, 2007.

[9] LK-Wf Grid Website. K-wf grid - home. Available online at http://www.kwfgrid.eu/. Last visited on: March 21st 2011.

[10] KIGForge Website. Owl tools: Projektinfo. Available online at https://kig.icsr.agh.edu.pl/projects/owltools/. Last visited on: March 21st 2011.

[11] WBEM Services Website. Wbem services. Available online at http://wbemservices.sourceforge.net/. Last visited on: March 21st 2011.

[12] FiVO Website. Kig/cim2owl - fivo. Available online at http://fivo.cyfronet.pl/trac/fivo. Last visited on: March 21st 2011.

[13] SourceForge Website. xcim2owl conversion tool. Available online at `http://sourceforge.net/projects/xcim2owl/`. Last visited on: March 21st 2011.

[14] Office of Government Commerce Website. Ogc - home. Available online at `http://www.ogc.gov.uk/`. Last visited on: March 21st 2011.

[15] Office of Government Commerce. *ITIL Lifecycle Publication Suite Books*. The Stationery Office, third edition, January 2007.

[16] itSMF International Website. itsmf international - it service management, itil and complimentary best practices. Available online at `http://www.itsmfi.org/`. Last visited on: March 21st 2011.

[17] DMTF Website. Home | dmtf. Available online at `http://www.dmtf.org/`. Last visited on: March 28th 2011.

[18] DMTF WBEM Website. Web-based enterprie management. Available online at `http://www.dmtf.org/standards/wbem`. Last visited on: March 28th 2011.

[19] DMTF CIM Website. Common information model. Available online at `http://www.dmtf.org/standards/cim`. Last visited on: March 28th 2011.

[20] DMTF Website. Cim & mof tutorial. Available online at `http://www.wbemsolutions.com/tutorials/DMTF/`. Last visited on: March 21st 2011.

[21] Andrea Westerinen and John Strassner. Common information model (cim) core model version 2.4. White paper, Distributed Management Task Force, August 2000.

[22] W3C RDF Website. Rdf primer. Available online at `http://www.w3.org/TR/rdf-primer/`. Last visited on: March 21st 2011.

[23] W3C OWL2 Website. Owl 2 web ontology language. Available online at `http://www.w3.org/TR/owl-overview/`. Last visited on: March 21st 2011.

[24] W3C SPARQL Website. Sparql query language for rdf. Available online at `http://www.w3.org/TR/rdf-sparql-query/`. Last visited on: March 21st 2011.

[25] W3C SWRL Website. Swrl: A semantic web rule language combining owl and ruleml. Available online at `http://www.w3.org/Submission/SWRL/`. Last visited on: March 21st 2011.

[26] John Hebeler, Matthew Fisher, Ryan Blace, Andrew Perez-Lopez, and Mike Dean. *Semantic Web Programming*. Wiley, Indianapolis, IN, 2009.

[27] Jorge E. López De Vergara, Víctor A. Villagrá, and Julio Berrocal. Semantic management: advantages of using an ontology-based management information meta-model. In *Proceedings of the HP Openview University Association Ninth Plenary Workshop (HP-OVUA'2002), distributed videoconference*, pages 11–13, 2002.

[28] Steve Battle and David Booth. The hp universal cmdb sparql adapter. In *HP Software Universe 2007*, 2007.

[29] Hibernate Website. Hibernate - jboss community. Available online at `http://www.hibernate.org/`. Last visited on: March 31st 2011.

[30] Kurt Rohloff, Mike Dean, Ian Emmons, Dorene Ryder, and John Sumner. An evaluation of triple-store technologies for large data stores. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *OTM Workshops (2)*, volume 4806 of *Lecture Notes in Computer Science*, pages 1105–1114. Springer, 2007.

[31] Jim Melton. Sql, xquery, and spa. Technical report, Oracle Corp., 2006.

[32] BSBM Website. Berlin sparql benchmark results - 02 22 2011. Available online at `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/results/V6/index.html`. Last visited on: March 28st 2011.

[33] Oracle Website. About solaris wbem services (solaris wbem developerś guide). Available online at `http://download.oracle.com/docs/cd/E19683-01/817-3096/chap-ov-23345/index.html`. Last visited on: March 28th 2011.

[34] Distributed Management Task Force. Cim operations over http version 1.3.1. Specification, Distributed Management Task Force, July 2009.

[35] Jim Davis, Paul Ferdinand, Carl Chan, Arora Ramandeep, and Dave Blaschke. The java api for web based enterprise management. Proposed final draft, WBEM Solutions, Inc., October 2009.

[36] SBLIM Website. Sblim main page. Available online at `http://sourceforge.net/apps/mediawiki/sblim/index.php?title=Main_Page`. Last visited on: March 28th 2011.

[37] openSLP Website. An api for service location. Available online at `http://www.openslp.org/doc/rfc/rfc2614.txt`. Last visited on: March 28th 2011.

[38] Jena Website. Jena - a semantic web framework for java. Available online at `http://openjena.org/`. Last visited on: April 2nd 2011.

[39] Sesame Website. openrdf.org ...home of sesame. Available online at `http://www.openrdf.org/`. Last visited on: April 2nd 2011.

[40] OWL API Website. The owl api. Available online at `http://owlapi.sourceforge.net/`. Last visited on: April 2nd 2011.

[41] HP Labs Website. Hp labs semantic web research. Available online at `http://www.hpl.hp.com/semweb/`. Last visited on: April 2nd 2011.

[42] Pellet Website. Pellet: Owl 2 reasoner for java. Available online at `http://clarkparsia.com/pellet/`. Last visited on: April 2nd 2011.

[43] Daniel J. Abadi, Adam Marcus, Samuel Madden, and Katherine J. Hollenbach. Scalable semantic web data management using vertical partitioning. In *VLDB*, pages 411–422, 2007.

[44] Daniel Alexander Smith, Alisdair Owens, M. C. Schraefel, Patrick Sinclair, Paul André, Max L. Wilson, Alistair Russell, Kirk Martinez, and Paul Lewis. Challenges in supporting faceted semantic browsing of multimedia collections. In *Proceedings of the semantic and digital media technologies 2nd international conference on Semantic Multimedia*, SAMT'07, pages 280–283, Berlin, Heidelberg, 2007. Springer-Verlag.

[45] Alisdair Owens, Andy Seaborne, Nick Gibbins, and mc schraefel. Clustered tdb: A clustered triple store for jena. November 2008.

[46] Mulgara Website. Welcome to the mulgara project. Available online at `http://www.mulgara.org/`. Last visited on: March 21st 2011.

[47] Ontotext Website. Owlim. Available online at `http://www.ontotext.com/owlim`. Last visited on: March 21st 2011.

[48] W3C Website. Largetriplestores - w3c wiki. Available online at `http://www.w3.org/wiki/LargeTripleStores`. Last visited on: March 21st 2011.

[49] Marian Babik and Ladislav Hluchy. On automated testing of description logic reasoners. In *Proceedings of the 5th International Conference on Distributed Computing and Internet Technology*, ICDCIT '08, pages 13–25, Berlin, Heidelberg, 2009. Springer-Verlag.

[50] Christian Bizer and Andreas Schultz. Benchmarking the performance of storage systems that expose sparql endpoints. In *In Proceedings of the ISWC Workshop on Scalable Semantic Web Knowledgebase*, 2008.

[51] Dennis Heimbigner. Dmtf - cim to owl: A case study in ontology conversion.

[52] W3C SPARQL Update Website. Sparql update. Available online at `http://www.w3.org/Submission/SPARQL-Update/`. Last visited on: March 21st 2011.

[53] JETM Website. Java(tm) execution time measurement library. Available online at `http://jetm.void.fm/`. Last visited on: March 21st 2011.

[54] Protégé Website. The protégé ontology editor and knowledge acquisition system. Available online at `http://protege.stanford.edu/`. Last visited on: March 21st 2011.

Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
BNF	Backus Naur Form
CIM	Common Information Model
CIMOM	CIM Object Manager
CMIP	Common Management Information Protocol
CMS	Configuration Management System
CMDB	Configuration Management Database
CRUD	Create, Read, Update and Delete
DMI	Desktop Management Interface
DMTF	Distributed Management Task Force
FRU	Field Replaceable Unit
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
ITIL	IT Infrastructure Library
JSR	Java Specification Request
MOF	Managed Object Format
OGC	Office of Government Commerce
ORM	Object-Relational Mapping
OWL	Web Ontology Language
RDBMS	Relational Database Management System
ROI	Return on Investment
SACM	Service Asset and Configuration Management
SBLIM	Standards Based Linux Instrumentation
SFCB	Small Footprint CIM Brooker
SLA	Service Level Agreement
SLP	Service Location Protocol
SNMP	Simple Network Management Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SWRL	Semantic Web Rule Language
TCO	Total Cost of Ownership
UML	Unified Modeling Language

W3C	World Wide Web Consortium
WBEM	Web Based Enterprise Management
XMI	XML Metadata Interchange
XML	Extensible Markup Language

Glossary

ARQ A query engine in Jena that provides SPARQL support.

Common Information Model Schema A collection of class definitions to represent managed objects.

Common Management Information Protocol A protocol that provides communication between network management applications and the management agents.

Common Information Model A hierarchical and object-oriented model to describe the components of a managed IT environment.

Common Information Model Object Manager Part of a CIM server that provides interaction between providers and management applications.

Configuration Item A Configuration Item represents any managed entity that is involved in providing an IT service.

Configuration Management Database A Configuration Management Database stores the Configuration Records of a managed entity through its whole life cycle.

Configuration Management System A Configuration Management System consists of a set of tools and multiple Configuration Management Databases that are used to manage the configuration data of an IT environment. This includes additional information about problems, incidents, changes and releases.

Configuration Record The details of a Configuration Item are stored in a Configuration Record during its life cycle.

Managed Object Any component of a computer system that is represented as a CIM Instance.

Managed Object Format The Managed Object Format is the language used to describe CIM classes.

Ontology An ontology defines the vocabulary that is needed to represent and describe a knowledge area.

Service Asset and Configuration Management An IT Infrastructure Library process that involves Configuration Management as well as Asset Management.

Service Level Agreement A contract between a service provider and a customer that defines the provided services and responsibilities.

SPARQL SPARQL is a query language for RDF developed by the W3C.

List of Figures

List of Tables

Listings

Appendix A

Installation Guidelines

The installation guideline is provided by the `README` file in the *SeConD-1.0.0 distribution.*

Appendix B

Contents of the CD

The CD contains following files and folders:

B.1 Files

Files located in the directory root:

thesis.pdf The Bachelor Thesis as PDF file.

thesis.ps The Bachelor Thesis as PS file.

abstract.pdf The abstract of the Bachelor thesis in English.

zusfsg.pdf The abstract of the Bachelor Thesis in German.

B.2 Folders

Folders located in the directory root:

Presentations The slides of the thesis presentation as PDF, PPSX, PPT and PPTX files.

Related Software Libraries and further software used in the thesis.

Related Work References A complete set of all referenced related work papers.

Thesis Latex Source Code LaTeX source code and images used in the thesis.

Thesis Software The source code distribution of SeConD-1.0.0.