A Domain Ontology for Designing Management Services

Ingo Pansa[1], Matthias Reichle[2], Christoph Leist[2], Sebastian Abeck[1]
Cooperation & Management
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
[1]{pansa, abeck}@kit.edu
[2]{matthias.reichle, christoph.leist}@student.kit.edu

Abstract— **Designing management systems based on service-oriented principles is a pragmatic approach to handle the challenges that distributed management is faced today. In order to conform to service-oriented principles, the elements of the management systems architecture – the management services – have to be designed along domain-specific concepts. Thus, modeling the domain IT Management becomes evident within service-oriented development processes. Considering existing approaches, domain modeling is not addressed explicitly, thus hampering the construction of management systems based on service-oriented principles. In this paper, we propose an ontology for the specification of the domain IT Management and present a refined development approach that enables the application of the presented ontology. The application of the ontology is demonstrated by designing management services for a standardized Incident Management Process.**

Keywords-management service; ontology; domain model

I. INTRODUCTION

With the adoption of web-based dynamic IT services by the business, management of distributed IT infrastructures becomes an evident part of a service provider's daily operations [22, 32]. Defining, designing and implementing management processes within the providers organization is therefore necessary. Automating these management processes is essential, as both reaction times can be limited and recurring errors by technical personnel involved in these processes are avoided [34]. Constructing a service-oriented solution built upon loosely coupled, process-oriented and reusable management services is desired. However, the challenges in realizing this are numerous [17, 21, 22, 23, 31]. From the perspective of an IT infrastructure operator, IT services are often created using a couple of different computing systems running different applications, interconnected by different networking technologies. In real life scenarios, often some hundred different management tools are utilized. Often these tools do not offer any standardized interfaces hampering the automation of management processes [24, 25].

In order to fully utilize the principles of service-oriented computing within the domain of IT Management, a clearly defined development process focusing the constructing of reusable management services is needed. This development approach has to consider both IT management standards and aspects of how to design service-oriented software solutions likewise. Although initial work discussing the application of service-orientation to construct management systems exists [6, 7, 21, 24, 25], little has been done to tackle this challenge on a conceptual level by explicitly focusing on the reusability and process orientation of flexible management systems.

In this paper, we deliver contributions to the addressed problems [2, 3] in that we clearly specify different aspects of models for designing reusable management services. The refined models are based on OWL ontology [4], enabling both the definition of a meta model and the application of the meta model to different management areas [36, 37]. Focusing an ontology-based definition of domain models not only supports the construction of reference models that can be shared among the scientific community [1, 9, 11, 14] but also the construction of concrete management tool support by the tool vendors [36, 37].

The remaining parts of this paper are structured as follows: Section 2 introduces related work and provides the background for applying ontological engineering for designing service-oriented systems. In Section 3, we introduce a refined development process for service-oriented design embedding an ontology-based meta model. Section 4 presents the contribution of this paper: we specify the conceptual meta model using an OWL ontology based on the refined definition of the necessary abstract syntax. In Section 5, we demonstrate the application of a reference model for designing management services for Incident Management that are both reusable and aligned to the management processes that they support. Finally, Section 6 concludes this paper and gives an overview of the work that is currently being done within our research group.

II. BACKGROUND AND RELATED WORK

Applying domain-driven techniques for designing management services that address the challenges discussed in [21, 22, 23, 32, 34], a revision of the approaches contributing to service-oriented management systems [6, 7, 24, 25, 42] becomes necessary. As we currently observe a shift towards web-based usage of dynamic IT Services ("Cloud Computing"), this holds even more. Standards such as ISO/IEC20000-1:2005 [19] only serve as a starting point, but lack of a foundation that is based on formalism thus

impeding the efficient development of flexible management systems.

As Erl [40] states, the analysis phase is the most important step in the service-oriented software development process towards well-designed services. Having a proper tool support by utilizing according modeling languages greatly influences the quality of the resulting analysis artifacts, as models have several advantages compared to informally defined analysis artifacts [12]. Following such structured development processes, analyzing, defining and modeling the domain for the desired information system serves as a starting point [8, 15]. Explicitly considering a certain domain greatly increases the possibility of engineered components to be reused [39].

As stated in [10], development teams tend to suffer from "UML fever" thus creating models that are too fine grained. While this is certainly true in general, considering analysis phases of typical development processes, both the support of modeling languages and the ability to describe the considered aspects in an intuitive way is desired likewise. Several recent works [1, 16] propose the use of ontology [18] for domain analysis mainly as a starting point leading to a meta model and later to a domain model [20, 33, 35]. Other authors [9, 13] even go one step further and use ontology as meta model or domain model itself. To unify both UML and ontology-based modeling within analysis phases, the OMG currently considers the definition of an overall UML-based meta model for ontologies, thus allowing to use several concrete syntaxes for ontology definition [29].

In our approach, this is exactly what we intend to do by utilizing the Web Ontology Language OWL [4].

III. DOMAIN-DRIVEN DESIGN

In order to be able to organize the different steps of a complex service-oriented software development project, different development process models have been proposed in the past (e.g., [5]). Their goals range from easy-to-adopt agile process to complex frameworks considering legal or cultural backgrounds of the involved parties. Focusing domain-driven design, several approaches have been discussed. While it is not our intend to present yet another development method, a basic understanding of the different steps required within service-oriented software development is required in order to comprehend and adopt the proposed domain ontology.

A. Overall development process

On a very abstracted level, software development is structured along different phases: based on requirements elicitation [41], the (possibly) informally defined requirements are analyzed and modeled using formally defined modeling languages. Two concerns are primarily considered within this analysis phase: structural analysis and dynamic analysis. While structural analysis deals with the definition of the domains, borders, stakeholders or objects, the dynamic analysis focuses the interaction of the elements that are identified within structural analysis. For both structural and dynamic analysis exists a couple of different models and modeling techniques, for which UML-based approaches can be regarded as the ones mostly preferred by software engineers. Due to the artifacts that are produced by the overall development process, the considered analysis phase is refined as service-oriented analysis. Different approaches are proposed to define necessary steps and development activities within service-oriented analysis [40].

Following service-oriented analysis, service-oriented design focuses on the definition of prescriptive models that clearly define the semantics of the to-be-implemented artifacts in means of the underlying platform concept. As a service-oriented software solution is desired, the underlying platform concept is bound to the principles of service-orientation (such as loose coupling, message-based communication, clear business relation).

B. Models

Based on the hitherto discussed development process, several models serve as a foundation to support the different development tasks. As we focus the overall analysis and design phases of the development process, we suggest no further assumptions on how to model the different artifacts that are produced during requirements elicitation. Typically, UML Use Cases are used to sketch up the desired functionality on a coarse granular level, supporting an understanding of both customers and business process analysts.

Focusing service-oriented analysis and service-oriented design, the central artifacts that are produced within these phases are domain models (analysis phase), service candidate models (according to Erl [40] in analysis phase) and service design models. To capture the overall choreography that different involved partners in a complex business scenario inhabit, business process models are used.

Due to the different semantics of each of the concrete models, different modeling languages are used. In Fig. 1, an overview of the different models and their interrelations is presented.

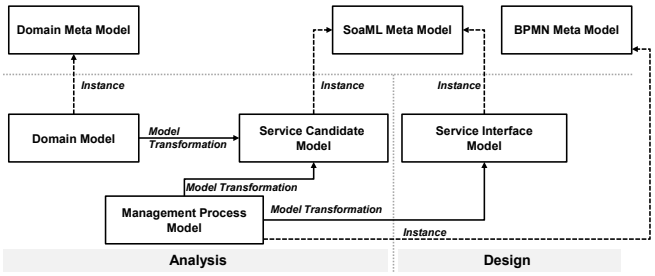

Figure 1. Different models in analysis and design phases

Both domain models and the meta model are defined using the Web Ontology Language (OWL) [4]. Although OWL originally was intended to be useful for semantically enriched resources in the World Wide Web, lately published work highlights the advantages of using both OWL to define project-specific analysis models and project-independent meta models. Business process models capture dynamic aspects of the structural elements that are modeled using domain models, wherefore both domain models and business

process models are used to derive service candidate models. Business process models can be defined using different languages from which one of the most accepted is the Business Process Modeling Language (BPMN), published by the OMG [28]. Considering the definition of service candidate or service interface models, we propose to utilize SoaML [30]. This upcoming standard for modeling service-oriented systems published by the OMG enables to define several aspects of service-oriented systems. One of the most interesting in our opinion is the concept of a service candidate, that defines required but not yet fully specified service capabilities. Based on the definition of service candidates, service interfaces can be engineered focusing the platform-specific requirements that service-oriented software systems are build upon.

C. Model Transformations

As domain models capture the structural aspects of the software system that is to be designed, using domain models to define the dynamic aspects prevents the involved stakeholders of defining the different elements with divergent semantics. Focusing on domain models being defined using OWL and process models being defined using BPMN, a simple transformation scheme can be defined as shown in Table 1.

TABLE I. DOMAIN META MODEL AND BUSINESS PROCESS META MODEL

Domain meta model element	Process meta model element
Management Area	Pool
Management Participant	Lane
Management Entity	Data Object
Management Basic Activity	Task
Management Composed Activity	Sub-Process

Although the transformation is defined informally, it proved that the process models we derived on the modeled OWL ontologies were much more intuitive to understand as they contained exactly the identified management participants, their executed activities and their required management entities.

IV. A DOMAIN ONTOLOGY

In addition to previous work we published so far [2, 3], an approach based on formal domain models has several advantages. Using OWL to define such formal semantics, in this section we outline the definition of a domain ontology based on our comprehensive abstract syntax.

A. Abstract syntax

The first step towards an ontology based on the requirements defined in ISO/IEC20000-1:2005 [19] is to gather the relevant vocabulary. In our previous work [2, 3] we derived the key concepts of the IT Management domain from the specification which are elaborated further with this paper. As a result, we identified certain concepts that can be used to model the IT Management domain in a functional centric way, providing the basis for the design of reusable and process oriented management services.

The central element of the domain is the *Management Area*, which represents one of the thirteen management processes like Incident Management or Change Management described in [19]. It holds all other entities related to a particular management process. Every Management Area consists of so called *Management Activity* elements which are used to denote a concrete activity performed to accomplish management goals. There are two specializations, namely *Basic Management Activity* and *Composed Management Activity*. A Basic Management Activity cannot be drilled down any further; it is atomic, whereas a Composed Management Activity is used to embody complex Workflows which consist of one or more Management Activities. In addition to the concepts of ISO/IEC20000-1:2005, we propose the modeling of *Management Capabilities*. A Management Capability is an abstract element that describes a certain capability which is used in Management Activities to fulfill their tasks. They can be refined to *Provided Management Capability* or *Required Management Capability*, depending on whether it is provided by a service e.g., a management tool or required by an activity, thus playing a major role in integrating existing functionality. To represent all necessary information required by a management process, we use *Management Entities*. Using the *Management Participant* concept, we model all actors involved in activities that belong to a certain Management Area. The last type of relevant information that needs to be modeled is a flexible way to describe requirements in regard of structure or handling of other elements in the domain, therefore we use *Management Policies*. One specialization of Management Policy is a *Management Entity Structure Policy* which specifies the structure an entity has to fulfill.

After identifying these basic concepts of IT Management, we have a solid foundation on which the IT Management ontology can be built on.

B. Ontology of the meta model

Using the OWL Web Ontology Language, we aim at a more formal specification of the key concepts introduced before. The specification heavily relies on the description of relations between OWL classes and restrictions regarding the validity of such relations between those elements. Furthermore, we define necessary conditions that elements of a specific class have to fulfill. Meta model elements (i.e. OWL classes) are not only restricted through abstract super classes but also inheritance of restrictions according to their parent class in the hierarchy of the OWL classes (see Fig. 2). Hereby, we obtain a framework describing how management areas can be modeled in a way that conforms to the meta model.

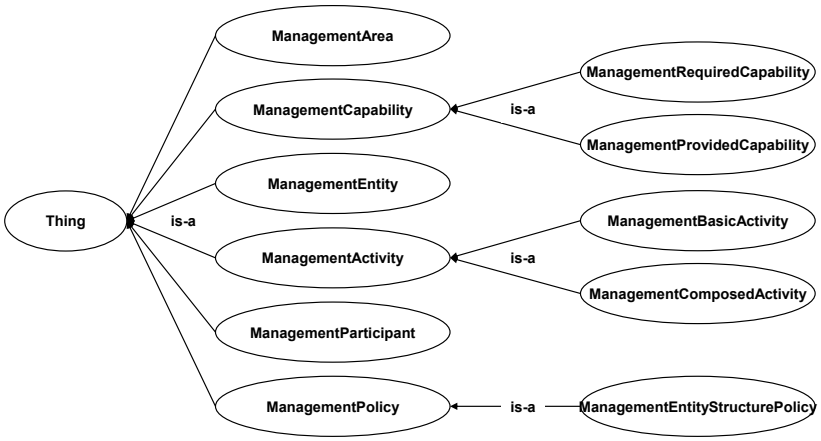

Figure 2. Taxonomy of IT Management

The associations between model elements are realized with OWL Object Properties. Based on [19], we identified six relations between model elements, where each one has an inverse. The relations below are depicted in the following style: *relation (~inverse)*.

Analogous to the central element in the meta model (Management Area) there is a corresponding central relation between the elements. The *contains (~isPartOf)* Object Property describes that a Management Area contains certain other elements, specifically those are Management Participant, Management Entity, Management Policy, Management Activity and Management Capability. To model information about various actors related to activities, we introduce *participatesIn (~hasParticipant)*. The fact that activities rely on capabilities to reach management goals is taken into account by the *requires (~isRequiredBy)* relation. As stated above, Management Activities may make use of Management Entities to gather or to store information. Therefore we use an Object Property called *operatesOn (~isOperatedBy)*. The entity itself is described in detail using a Management Entity Structure Policy that is bound to the Management Entity by *defines (~isDefinedBy)*. The sixth pair of relations models the composition of Management Activities. To represent the concept of composition, we use *isComposedOf (~isUsedBy)*.

```
(contains some ManagementActivity)
 and (contains some ManagementParticipant)
 and (contains only
     (ManagementActivity
     or ManagementCapability
     or ManagementEntity
     or ManagementParticipant
     or ManagementPolicy))
```

Figure 3. OWL Restrictions for Management Area class

The relations introduced in the paragraph above in conjunction with the key concepts identified earlier are now used to further specify the demands we make on domain models of IT Management. This is achieved through heavy use of OWL Restrictions, which can be seen in the excerpts of our OWL definition above (Fig. 3).

The Management Area element may hold the elements depicted earlier but to be a valid Management Area in our understanding it has to contain at least one Management Activity as well as a corresponding Management Participant. This results in the following OWL Restrictions for Management Area.

Every individual inside or every class inheriting of the Management Area class thus has to satisfy those requirements. To specify the nature of Management Activities, we restricted the members of the Management Activity class as shown in Figure 4.

```
(isPartOf only ManagementArea)
 and (isPartOf exactly 1 ManagementArea)
 and (hasParticipant some
ManagementParticipant)
 and (hasParticipant only
ManagementParticipant)
 and (operatesOn some ManagementEntity)
 and (operatesOn only ManagementEntity)
 and (requires some ManagementCapability)
 and (requires only ManagementCapability)
 and (ManagementBasicActivity or
ManagementComposedActivity)
 and (isUsedBy only ManagementComposedActivity)
```

Figure 4. OWL Restrictions for Management Activity class

The code presented in Fig. 3 and Fig. 4 reflects the fact that each Management Activity belongs to exactly one Management Area and it has to have at least one participant. Furthermore, we specified the necessity of a corresponding Management Entity as well as one or more Management Capabilities that are used by it. The last two statements say that each activity is either a basic or a composed activity and if it's composed, it can only be used by another Management Composed Activity.

All the other elements of the IT Management ontology are formalized in the same manner, resulting in a rather complex description of concepts and their interrelations in the IT Management domain as seen in Fig. 5.

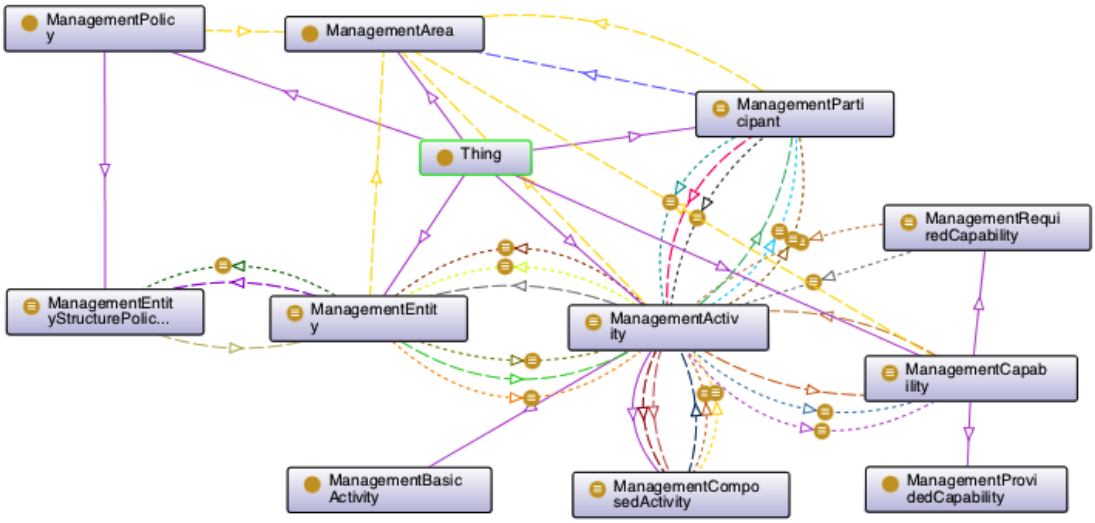

Figure 5. Domain ontology for IT Management

Based on this meta model, we now intend to describe the thirteen management processes mentioned in ISO/IEC20000-1:2005.

V. Designing Management Services for Incident Management

The following application example is divided into two parts: Part 1 deals with the definition of the domain ontology for Incident Management while part 2 demonstrates the design of concrete management services based on the proposed domain ontology.

A. Ontology for Incident Management

The domain ontology for the specific Management Areas is constructed by extending the OWL ontology of the meta model following some basic rules, the most important one being not to introduce classes directly under the Thing root class. Hereby, all classes of the domain model are restricted by the definitions of the meta model elements.

As a preparatory step, we need to identify the relevant elements that are needed to model the desired management process, which is described in detail in our previous works [2, 3]. As an example, we identified the elements shown in Table 2 for the Incident Management Process.

TABLE II. Elements for Incident Management Ontology

Meta Model Element	Domain Model Element
Management Area	Incident Management
Management Participant	Incident Manager, ServiceDeskEmployee, Specialist
Management Entity	Entity dedicated to Incident Management: Incident Record Further entitites: Known Error Record, Workaround Record, Configuration Management Database Record (CMDB Record)
Management Activity	Record Incident, Determine Business Impact, Prioritize Incident, Classify Incident, Escalate Incident, Resolve Incident, Inform Customer, Create Incident Record, Update Incident Record
Management Policy	Incident Record Structure Policy

The aforementioned elements now can be specified through further restrictions in line with the ones being specified at the meta model layer of the ontology. This leads to a well described model of the Incident Management Process according to ISO/IEC20000-1:2005 and valid in respect of the meta model, as seen e.g., in the fragment of our ontology below (Fig. 6), covering details of the Management Area Incident Management. These restrictions are added to the ones inherited through the hierarchy which places Incident Management beneath Management Area.

```
(contains some CreateIncidentRecord)
 and (contains some
CreateIncidentRecordCapability)
 and (contains some IncidentRecord)
 and (contains some
IncidentRecordStructurePolicy)
 and (contains some RecordIncident)
 and (contains some ServiceDeskEmployee)
```

Figure 6. OWL Restrictions for Incident Management area

For instance, we can see that Incident Management contains an element called Create Incident Record that actually is a Management Activity belonging to the said Management Area. On the other hand the definition of Create Incident Record contains the inverse of that information as well as other relations to further elements.

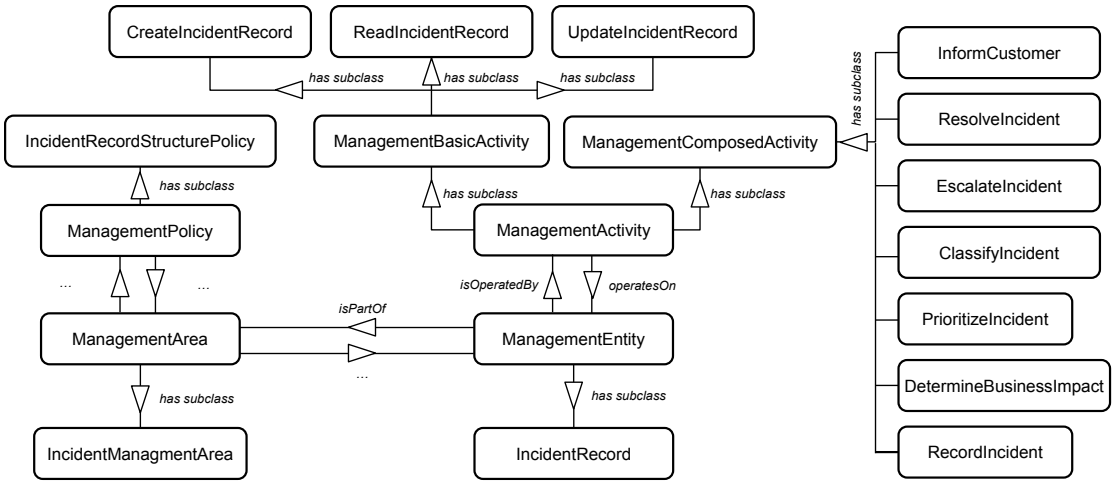

Figure 7. Domain ontology for Incident Management

By describing each element of Table 2 in the way seen above, we achieve a domain ontology for Incident Management forming a reference model according to Fig. 5. Fig. 7 shows a graphical excerpt of the ontology for Incident Management.

B. Incident Management Service Design

With the definition of ontology for Incident Management, the design of management services supporting automated management processes can be based on a solid foundation that transforms functional requirements to executable code. Following the overall development process as presented in Section 3, in this section we briefly give an overview of how to leverage the benefits of ontology-based domain-driven software development focusing management services.

Following the rules presented in [2] and taking into account the domain-knowledge formally specified in the OWL ontology, we are now able to construct management services in a comprehensible and repeatable way using SoaML [30]. One of the rules states that for every Management Entity a SoaML Message Type object should be created. Considering the Incident Management Process, we identified only one element residing directly beneath the Management Entity class in the OWL-hierarchy. Therefore we model a SoaML Message Type with the name of Incident Record and the structure defined in the Management Entity Structure Policy referenced through the isDefinedBy Object Property. Another principle we propose is to introduce a SoaML Capability that groups all Management Capabilities of the domain ontology that handle a single Management Entity. This results in a SoaML Capability named Incident Record Service which contains the operations according to the Management Capabilities of the OWL ontology, namely Create Incident Record, Read Incident Record and Update Incident Record. By applying some more rules as the ones we are able to present within the scope of this work, we are able to identify the most important SoaML model elements based on elements of our OWL ontology and their position inside the hierarchy, which places every element of the domain below one specific element of the meta model layer.

The SoaML results of the procedure described above can be seen in Fig. 8 showing SoaML Capabilities and Service Interfaces.

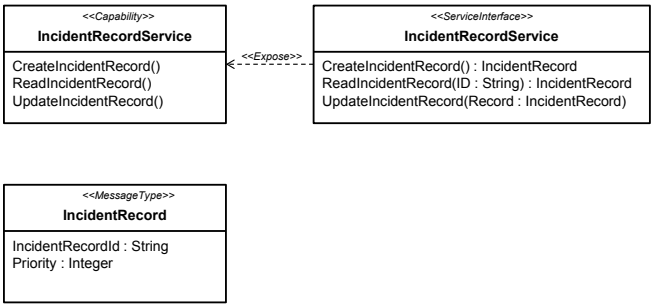

Figure 8. SoaML Capability and Service Interface reflecting domain concepts

Through further steps the SoaML Service Interfaces can be refined and finally realized by implementing Web services using technologies like WSDL [38] or its semantic annotated sibling SAWSDL [26] (see Fig. 9).

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="IncidentRecordService"
[...]
xmlns:sawsdl="http://www.w3.org/ns/sawsdl">
[...]
<xsd:complexType name="IncidentRecord"
      sawsdl:modelReference="http://domp.cm-
tm.uka.de/ontologies/ITSMOntology#IncidentRecor
d">
   <xsd:sequence>
      <xsd:element name="IncidentRecordId"
       type="xsd:string"></xsd:element>
      <xsd:element name="Priority"
       type="xsd:int"></xsd:element>
   </xsd:sequence>
</xsd:complexType>
[...]
```

Figure 9. SAWSDL excerpt showing references to the domain ontology

As the application example demonstrates, the semantics of the modeled domain artifacts can be preserved into the subsequent design steps. This finally helps to close the gap between analysis and design phases thus lowering development efforts in round trip engineering.

VI. CONCLUSION

For tackling the complexity that distributed management is faced to today, the automation of management processes is evident. Building management systems upon loosely coupled management services strongly supports this approach. However, several aspects have to be considered, as service-oriented design and analysis is a creative software development issue in which several different stakeholders are involved. For being able to model requirements and support the development process using model-based transformation techniques, analyzing and understanding the domain is essential. This paper addresses this situation and delivers several contributions.

In Section 3, a refined development process for service-oriented analysis and design was presented. The development process is generic in a way that concrete process models for software development are abstracted, but yet essential roles, stakeholders and artifacts are defined. The presented development process defined basic steps and activities that utilize ontological engineering and therefore served as a foundation for a concrete application of our approach in further scenarios or projects. Section 4 introduced a formalized meta model of the domain IT Management that served as a foundation for the definition of a reference model for Incident Management. The presented meta model focused problems that were previously addressed by our research group [2, 3]. Using OWL to define the domain ontology based on formal semantics not only allows to construct tool support that directly guides involved stakeholders during initial analysis or design activities but also to consider service design quality by the application of metrics suites. Section 5 demonstrated the application of the development process. We showed that utilizing ontological engineering supports the construction of management services that align with certain design principles, of which we mostly addressed in this paper the general requirement of a clear process-alignment of designed services.

Considering the experiences that we made within the development project, further work is necessary. Based on the development method, the OWL-based domain meta model and the OWL-based Incident Management reference model, introducing automated metrics applications of concrete service designs in regard of several service design quality aspects seems to be possible. For instance, semantically enriched service models would allow automated classification based on the modeled management area context, thus leading to possible better reusability if future requirements are slightly changing. Considering the evolution of the proposed ontology, both an ontology repository and corresponding tools are necessary.

REFERENCES

[1] D. Gasevic, D. Djuric, and V. Devedzic, Model Driven Engineering and Ontology Development, Springer, Heidelberg, 2006.

[2] I. Pansa, F. Palmen, S. Abeck, K. Scheibenberber, “A Domain-driven Approach for Designing Management Services”, SERVICECOMPUTATION2010, Lisbon, 2010.

[3] I. Pansa, P. Walter, K. Scheibenberger, and S. Abeck, “Model-based Integration of Tools Supporting Automatable ITSM Processes”, IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS Wksps), 2010, Page(s): 99 – 102.

[4] World Wide Web Consortium (W3C), Web Ontology Language (OWL), W3C Recommendation, 2004.

[5] A. Arsanjani, S. Gosh, A. Allam, T.Abdollah, S. Ganapathy, and K. Holley, „SOMA : A method for developing service oriented solutions,“ in IBM Systems Journal, Vol. 47 (3), pp. 377-396, 2008.

[6] G. Aschemann and P. Hasselmeyer, „A Loosely Coupled Federation of Distributed Management Services“, Journal of Network and Systems Management, Vol 9, No. 1, 2001.

[7] N. Anerousis, „An Architecture for Building Scalable, Web-Based Management Services“, Journal of Network and System Management, Vol. 7, No 1., 1999.

[8] G. Arango, Domain Analysis - From Art Form To Engineering Discipline, ACM SIGSOFT Software Engineering Notes, Volume 14, Issue 3, 1989.

[9] U. Aßmann, S. Zschaler, and G. Wagner, „Ontologies, Meta-models and the Model-Driven Paradigm“, Ontologies for Software Engineering and Software Technology (2006), pp. 249-273, 2006.

[10] A. E. Bell: Death by UML Fever, ACM Queue, Volume 2 Issue 1, March 2004, 2004.

[11] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, „What are Ontologies, and why do we need them?“, IEEE Intelligent Systems and their Applications, Vol. 14, Issue 1, pp. 20-26, 1999.

[12] G. Cernosek, and E. Naiburg, „The Value of Modeling,“ IBM Developer Works Whitepaper, 2004.

[13] V. Devedzic: Understanding Ontological Engineering, Communications of the ACM, Vol. 45 (4), pp. 136-144, 2002.

[14] R. de Almeida Falbo, G. Guizzardi, and K. C. Duarte, „An Ontological Approach to Domain Engineering“, Proceedings of the 14th international conference on Software engineering and knowledge engineering (SEKE2002), 2002.

[15] X. Ferré, and S.Vegas, „An Evaluation of Domain Analysis Methods“, In 4th CAiSE/IFIP8.1 International Workshop in Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD99), 1999.

[16] T. Stahl, M. Völter, S. Efftinge, and A. Haase, Modellgetriebene Softwareentwicklung, dpunkt.verlag, 2007.

[17] S. D. Galup, R. Dattero, J. J. Quan, and S. Conger, „Information Technology Service Management: an emerging area for academic research and pedagogical development“, Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, pp.52, 2007.

[18] T. Gruber, „A translation approach to portable ontology specifications“, Knowledge Acquisition, Vol. 5, Issue 2, pp. 199-220, 1993.

[19] ISO/IEC20000-1:2005, Information technology – Service management – Part1: Specification, International Standards Organization (ISO), 2005.

[20] G. Kappel, E. Kapsammer, H. Kargl, G. Kramler, T. Reiter, W. Retschitzegger, W. Schwinger, and M. Wimmer, „Lifting Metamodels to Ontologies“, Lecture Notes in Computer Science, Volume 4199/2006, pp. 528-542, 2008.

[21] P. Kumar, „Web Services and IT Management“, ACM Queue, Volume 3 Issue 6, 2005.

[22] V. Machiraju, C. Bartolini and F. Casati, „Technologies for Business-Driven IT Management“, in Extending Web Services Technologies: the Use of Multi- Agent Approaches, Kluwer Academic, 2004.

[23] A. Moura, J. Sauve and C. Bartolini, „Research Challenges of Business-Driven IT Management“, 2nd IEEE/IFIP International Workshop on Business- Driven IT Management, pp.19-28, 2007.

[24] C. Mayerl, F. Tröscher, and S. Abeck, „Process-oriented Integration of Applications for a Service-oriented IT Management“, The First International Workshop on Business-Driven IT-Management, BDIM'06, pp. 29-36, 2006.

[25] C. Mayerl, T. Vogel and S. Abeck, „SOA-based Integration of IT Service Management Applications“, 2005 IEEE International Conference on Web Services (ICWS 2005), 2005.

[26] World Wide Web Consortium (W3C): Semantic Annotations for WSDL and XML Schema, W3C Recommendation, 2007.

[27] Organization for the Advancement of Structured Information Standards (OASIS), Reference Model for Service Oriented Architecture, Version 1.0, OASIS, August 2006.

[28] Object Management Group (OMG), Business Process Model and Notation BPMN, Version 2.0, OMG,Januar 2011.

[29] Object Management Group (OMG), Ontology Definition Metamodel (ODM). Version 1.0, OMG, Mai 2009.

[30] Object Management Group (OMG), Service-oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS), FTF Beta 1, OMG, April 2009.

[31] G. Pavlou, „On the Evolution of Management Approaches, Frameworks and Protocols: A Historical Perspective, Journal of Network and Systems Management“ Springer New York, Vol. 15, Issue 4, pp. 425- 445, 2007.

[32] J. Sauvé, A. Moura, M. Sampaio, J. Jornada and E. Radziuk, „An Introductory Overview and Survey of Business-Driven IT Management“, 1st IEEE/IFIP International Workshop on Business-driven IT Management (BDIM 2006), 2006.

[33] S. Staab, T. Walter, G. Gröner, and F. S. Parreiras, „Model Driven Engineering with Ontology Technologies“, Lecture Notes in Computer Science, Volume 6325/2010, pp. 62-98, 2010.

[34] V. Tosic, „The 5 C Challenges of Business-Driven IT Management and the 5 A Approaches to Addressing Them“, The First IEEE/IFIP International Workshop on Business-Driven IT Management, 2006.

[35] M.-N. Terrasse, M. Savonnet, E. Leclercq, T. Grison, and G. Becker, „Do We Need Metamodels AND Ontologies for Engineering Platforms?“, Proceedings of the 2006 international workshop on Global integrated model management, 2006.

[36] J. E. López De Vergara, V. A. Villagrá, J. Berrocal: Semantic Management: advantages of using an ontology-based management information meta-model, Proceedings of the HP Openview University Association Ninth Plenary Workshop (HP-OVUA'2002), 2002.

[37] J. E. López De Vergara, V. A. Villagrá, J. Asensio, J. Berrocal: Ontologies Giving Semantics to Network Management Models, IEEE Network, Vol. 17, pp. 15-21, 2003.

[38] World Wide Web Consortium (W3C), Web Service Description Language (WSDL) Version 2.0 Part1 Core Language, W3C Recommendation, 2007.

[39] J. Wang, J. Yu, P. Falcarin, Y. Han, and M. Morisio, „An Approach to Domain- Specific Reuse in Service-Oriented Environments“, Proceedings of the 10th international conference on Software Reuse: High Confidence Software Reuse in Large Systems, 2008.

[40] T. Erl: SOA, Principles of Service Design, Prentice Hall, 2008.

[41] B. Bruegge, and A. H. Dutoit, Object-Oriented Software Engineering Using UML, Patterns and Java, Pearson Education, 2009.

[42] G. Tamm and R. Zarnekow, „Umsetzung eines ITIL-konformen IT-Service-Support auf der Grundlage von Web-Services“, Wirtschaftsinformtik 2005, pp. 647-666, 2005.