A Domain-driven Approach for Designing Management Services

Ingo Pansa[1], Felix Palmen[2], Sebastian Abeck[1]
Cooperation & Management
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
[1]{pansa, abeck}@kit.edu
[2]felix.palmen@cm-tm.uni-karlsruhe.de

Klaus Scheibenberger
IT Infrastructure and Services
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{scheibenberger}@kit.edu

***Abstract*— A service-oriented software solution to flexibly support changing business environments requires the existence of an adaptable management support system. Decoupling management processes from concrete tools by encapsulating needed management functionality into management services can help meet this requirement. However, creating management solutions is a difficult and challenging task. Due to the complexity of domain management, a formal approach based on a domain model and assorted design rules would help to increase the stability of engineered solutions. Existing approaches tend to gather only at the tools level, while neglecting process requirements, which result in solutions that are hard to adapt. In this paper, we discuss the value of domain modeling to address this situation and demonstrate how designing management services by using a set of assorted design rules can be achieved. This approach is exemplified within a concrete incident management scenario.**

Keywords- domain-driven design; service design; management service; incident management

I. INTRODUCTION

With the shift towards service-oriented computing, a decoupling of needed functionality and enabling implementation has been reached. While the functionality that is needed today is derived from business requirements, enabling implementation is bound to technology. This has led to a decoupling of technology-independent business processes and technology-dependent IT systems. Business processes can now be adapted to changing requirements more easily. For instance, adding a refined debit check to typical invoice processing can now be formulated in terms of financial semantics rather than in terms of technological attributes, because needed functionality can be added by searching for debit services rather than technology-bound debit calculating software components.

Considering these extended possibilities, operational support of these services has to be adaptable as well. The challenges in realizing this are numerous [2, 15, 23]. From an IT infrastructure perspective, IT services are often created using a vast number of different computing systems running various different applications, which are interconnected by using different networking technologies. It seems all but impossible to use or create one single management tool considering all the different vendor technologies or fulfilling the special requirements that IT organizations typically have. From the perspective of a management tools supplier, it seems like it is difficult for them to ship their tools with open and generically applicable information and function models to operate the various different component technologies, because several different approaches for describing management information exist.

To increase the complexity even further, IT organizations have now started to restructure their management activities to align with best practices or standard proposals that can be derived from approaches such as ITIL [17] or ISO20000 [3]. While integration problems on the technical level within the domain of IT management always existed due to heterogeneous environments, the alignment with management processes that has been adopted lately requires the introduction of completely new tools – or at least to extend the ones that exist with the functionality to support the adoption of process requirements. Applying service-orientation in solving these issues seems to be a pragmatic, yet powerful way to both integrate existing tools and management infrastructures, as well as to align with management processes. Therefore service-orientation not only promotes adapting to new requirements more easily, but also the ability to reuse existing components.

In order to utilize the principles of service-oriented computing and service-oriented architecture to solve these challenges in managing IT services, a clearly defined development approach is needed. This development approach has to consider today's standards and best practices, as well as how to integrate existing management tools . Although some work that uses the application of service-orientation to construct management systems that are based on loosely coupled management services exists [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], little has been done to tackle this challenge on a conceptual level by focusing on the reusability and adaptability of future management systems.

This paper proposes an improved meta-model for the domain of process-oriented IT Management and an approach to applying the meta-model for the purpose of constructing reusable and adaptable management services. The approach is presented in the form of rules that allow a repeatable development method. Focusing on a domain meta-model that is built on the requirements of process-oriented standards enables constructing services that are aligned with the processes that the services are intended to support. Furthermore, these services serve as a central point for constructing integrative adapters to existing management tools. A formalized model enables the construction of development supporting tools, therefore the design of the management services system can be performed based on computational support. In our opinion, understanding the semantics of the terms the domain IT management is faced with is crucial in the construction of such a service-oriented management system. Therefore, having modeled the structure of the domain IT management for the purpose of deriving management services is a fundamental part.

The remaining parts of this paper are structured as followed: Section 2 introduces related work and provides the background for constructing a management platform that is built on service-oriented principles. In Section 3, we discuss the value of modeling the domain for a proposed solution and present an extension to the domain meta-model presented in [1]. Section 4 presents the benefits of this paper: We embed a domain-driven and rule-based development method into a typical software development process, demonstrate domain modeling applied to a typical management activity performed within incident management and introduce an assorted set of rules to support the domain-driven derivation of management services. Section 5 describes our experience with a prototypical implementation, where we applied the proposed method within a real world scenario. 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

The efforts of recent years to structure and tackle the complexity that management solutions are faced with have led to standard specifications for the definition of IT service management processes. The most prominent representative is ISO20000-1:2005 [29]. This particular standard definition presents a taxonomy introducing minimum functional requirements that implementations of different management solutions have to fulfill. While ISO20000-1:2005 is mainly based on the best practice suggestions presented by the Information Technology Infrastructure Library (ITIL, [16]), a clear and formal representation of the proposed entities, activities or participants, is still missing. Nevertheless, since [29] introduces the elements of the domain, our aim is to construct software solutions, where the standard definition serves as one input for the creation of a commonly accepted ontology, with which a formalized meta-model could be developed.

Applying service-orientation to solve integration issues (adaptability, reusability) is assumed to be one feasible approach [2, 17, 22]. Based on the suggested best practices of the ITIL [16], much research focusing on the construction of service-oriented management solutions has been done. Tamm and Zarnekow [12] derive web services from a typical definition of an incident management process, but neglect the domain as part of this process. It seems difficult to give any statement regarding the adaptability or reusability of the solution that they present.

Mayerl and Abeck et al. focus the integration of existing management tools along process-oriented management scenarios [14, 15]. Although a systematic development method is proposed, neither domain modeling nor specific rules for designing management services are discussed. The approaches presented in [14, 15] are rather general and do not consider formal aspects. Furthermore, standard requirements are neglected. In [13], an automated management process is implemented based on web services, but both a structural analysis, and a systematic method are missing.

Aschemann and Hasselmeyer deal with the principles of a service-oriented architecture in supporting management systems [4]. While both domain modeling and systematic development methods are missing, at least some architectural guidelines can be concluded from their work. For instance, different components enabling communication between management services are needed. Furthermore, some functionality enabling the location and lookup of existing management services is also needed. Anerousis discusses an architecture for building scalable management services [5], however it lacks formalization of the given domain. Lu et al. examine management services on the managed resource level [7, 8, 9], but not a systematic and overall development method, which integrates both process and resource requirements. Standard specifications such as WS-Management [10] or WSDM-MUWS [11] only deal with the managed resource level.

Different approaches for introducing a service taxonomy have been suggested [18, 19]. Based on these ideas, we have observed that it is possibile to clearly distinguish different types of management services, which is why we refer to management basic services and management process services when clarifying different characteristics of these services.

Our proposed design method of applying some assorted rules for deriving services is similar to the one presented in [20], but extends it by capturing not only elements of business process models, but also model instances of an overall domain analysis model. A good overview of different approaches for domain analysis can be found in [21]. The authors argue that although many different approaches for constructing domain models exist, software systems that support different problem domains differ in many aspects, which is why there is no existing modeling approach that is suitable for every kind of scenario. Based on their evaluation results, our approach is based on functional decomposition using rule support for creating both domain models and designs for management services.

III. A Domain Met- Model for IT Management

Describing the semantics of the elements of a domain serves as a building block for developing a software solution that is tightly aligned to its requirements. In this section, we give a summary of our motivation for using domain modeling and present initial and ongoing work in the area of development tool support for creating standardized and formalized models for services within the domain IT Service Management.

A. *The value of domain modeling in IT Management*

As the process of creating software systems becomes more complex, formal descriptions are required to engineer these systems. This can be saidfor any of the disciplines, from eliciting the requirements that a solution has to fulfill to creating detailed models describing the structure of the software architecture or aspects concerning control flow. However, formal descriptions are hard to achieve. One building block is the description of the semantics of the single elements of the domain that the desired solution will use. Identifying this information in relation to similar problems leads to a classification schema, which can easily be reused. This is referred to as domain modeling. Domain modeling is a pragmatic approach utilizing modeling techniques that are well understood. Creating a domain model has several advantages, one of the most important ones might be the fact that software systems derived from domain models show a higher degree of reusability if extensions to these software systems are created from the same domain models.

Nevertheless, a domain model has to be abstracted in some way to really be adaptable. Therefore, we decided to create a domain model that is based on ISO20000-1:2005 [29], which was enriched with some of the typical patterns of the Information Technology Infrastructure Library [16] best practices and existing process modeling approaches, for instance a separation of atomic and composed activities can be found in the Workflow Management Coalition Meta-Model [30]. Using such a domain model allows designing management services that are tightly aligned with the domain that the services are intended for. Furthermore, automated design evaluators can be constructed measuring the overlap of domain model instances and the instances of service models, thereby allowing for automated design decision support.

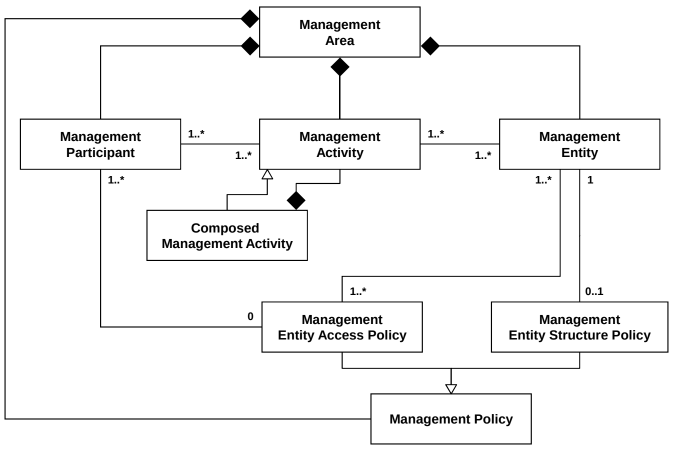

Figure 1. Conceptual Meta model of the Domain IT Management

A domain analysis method always consist of two things [21]: an ontology along with a taxonomy of this ontology defining a meta-model of the domain, and a process that allows the construction of model instances of this domain. The concepts of the Domain Meta-Model have been introduced in [1]. (see Figure 1). A Management Area contains Management Participants, Management Activities, Management Entities and also Management Policies, which can be refined to policies to define access to specific entities or policies to define the structure of entities. Management Activities can be refined to Composed Management Activities, e.g. a Management Area defines one single functional area such as Incident Management. Aiming at defining services aligned with models instantiated from this meta–model. Both the modeling element Management Activities and Management Area can be considered to offer capabilities which are independent of concrete realizations. These capabilities are later used to derive management services. To capture this aspect within the meta-model, the meta-model is simply extended as depicted in Figure 2.

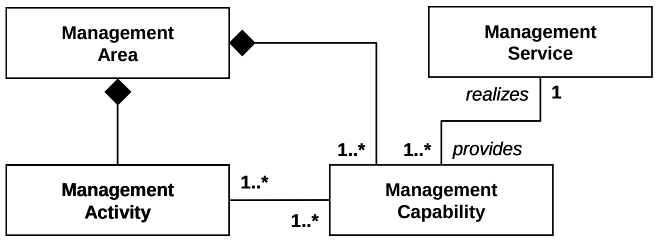

Figure 2. Extension for Deriving Management Services

Based on this domain meta-model and its extension, an assorted set of rules underpinning our development method is presented within the next section.

B. *Rule-based derivation of management services*

Focusing a refined derivation of management services by considering design decisions dedicated to the domain of IT management, a discussion of specific elements of the meta-model is necessary. The meta-model used so far provides elements for modeling management capabilities and the conceptual management services providing them. Those services are needed as part of the domain model because by considering them as elements of the domain, a clear semantic relation to the management capabilities and management activities is possible.

Of course, the sources used to model the domain do not mention management services because these are not part of the plain management view, but added to the domain by the decision to support management processes with a service-oriented architecture. Therefore, since adding management activities, management areas, management participants, etc. to the model directly from the text of e.g. ISO20000-1:2005 is a straightforward process, modeling management capabilities and management services involve design decisions. Therefore, in order to achieve reproducible results when modeling the domain, derivation rules are needed for naming and coupling management capabilities with management services. This repeatability is needed to preserve the value of the domain model, such as the increased degree of reusability in the resulting service design.

The conceptual management services given in the domain model do not yet support an implementation of a service-oriented architecture. We still need a service model that describes the services in detail with all service operations and their signatures. For such a service model, a specialized UML-derived modeling language like SoaML

[22] is a natural choice. Again, a requirement for the service model is repeatability, so we will introduce more derivation rules for the transition from the domain model to a SoaML model of the final service interfaces.

In order to maximize the benefit of our approach, these rules must be defined carefully. One important factor is consistent naming, so that the semantics of a service and its operations can be understood by looking at the domain model. Therefore, rules for naming are strict and do not leave room for individual decisions. Another aspect to keep in mind is reusability of individual management services. By defining rules that result in the modeling of exactly one service per management entity, the resulting services have few interdependencies and a single service can easily be exchanged, for example by using an adapter for an existing management tool.

IV. Applying Domain Modeling for Desiging Management Services

For demonstrating the value of domain modeling, a concrete scenario is presented. The next sections address a typical incident management process that serves as an input artifact for deriving adaptable yet business-aligned services. Although our approach refers to the term management process, a structural analysis of the domain is performed that not only take the dynamic parts of a process into account, but also the static relationship of domain elements involved within this process. Following this approach, the derived services can be reused in further development efforts if extensions to an existing system are necessary.

A. Overall development method

Since structured development methods for the purpose of deriving and designing a service-oriented software solution are common today, we briefly describe the necessary steps to perform in order to create a set of management services that are aligned with a model of the domain.

First of all, an analysis of the standard specification for process-oriented IT Service Management ISO20000 leads to an overview of the activities, entities and participants that constitute the management capabilities of one management area, for instance incident management. Adding policies (entity structure policies and entity access policies), the elements of an overall domain model are given.

Within the next step, these domain elements are modeled. To avoid misconceptions of the relationships of these elements with each other, a formal meta-model is needed that clearly defines the syntax and semantics of each single domain element. Such a meta-model can be found in [1].

As an improvement to the approach presented in [1], the derivation of services is performed using an assorted set of rules. These rules take several aspects of a model-to-model transformation of an instance of the domain model to an instance of a model of management services into account. Since the design of services is a highly complex task, in which several design decisions have to be made, we propose a two-step approach, stemming from domain models to a model of service candidates and finally a model of technology-independent service interface descriptions. Introducing such a two-fold step enables grouping several service candidate operations into combined service definitions if a service within a given management system already exists.

B. The Incident Management Process

The Incident Management Process is one of the critical processes dealing with service disruptions. Incident Management is established in nearly each service provider's organization offering defined IT services to customers utilizing IT services for supporting IT-based business processes. Since customers are directly faced with incident management for service failures, providers are interested in controllable execution of this process. As business requirements change requirements for supporting services management support and flexible management components are needed. According to [24], the Incident Management Process has a high recurrence rate and a high organizational structure. This process is mostly well suited to workflow support, which we intend to realize using a service-oriented architecture.

ISO20000:1-2005 defines the objective of the incident management process as the ability "to restore agreed upon services as fast as possible or to respond to service requests" [3]. In analyzing the definition of incident management, the following elements of the domain model can be identified:

Entities: Incident Record, Workaround Record,

Participants: First Level Support, Second Level Support

Activities: manage impact, record incidents, prioritize, determine business impact, classify, update, escalate, resolution and formal closure of all incidents.

Policies: Incident Entity Structure Policy, Incident Entity Access Policy

Having identified these elements, a formal model of the domain can now be constructed. For the sake of simplicity, we will look at the assorted management activity Prioritize Incident that is used to determine the impact of a service failure and to add a priority value to the related incident record.

C. Modeling one Management Activity

Modeling of a management activity in the ITSM domain model is done in several steps. First, sub-activities are identified from the ISO20000-1:2005 text. The definitions of other management processes are also looked at in order to pick up interconnections with other processes. In the second step, known patterns and principles of management architectures, such as OSI management [25], WBEM [26], etc., are considered in order to find matching sub-activities that directly model the usage of functionality provided by existing components. Finally, the management capabilities needed to perform these management activities are modeled and the conceptual management services providing these capabilities are defined.

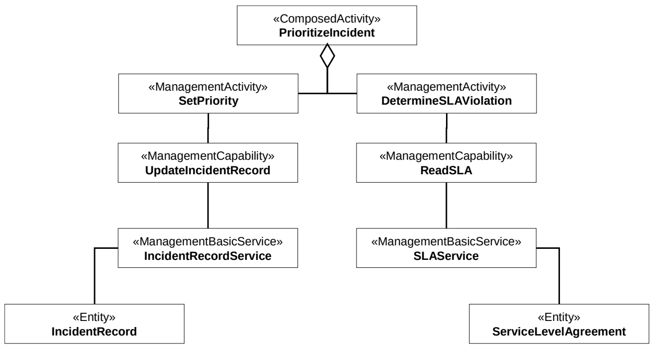

Figure 3. Management Activity PrioritizeIncident with process-related sub activities

In order to achieve a consistent model, rules are applied. The notation [Entity] in these rules puts the name of the entity instance here.

(Rule 1) Management capabilities that create, read or update an entity are named as follows: Create[Entity], Read[Entity] and Update[Entity].

(Rule 2) All management capabilities operating on an entity are provided by a single management service called [Entity]Service.

(Rule 3) Management capabilities that communicate an entity to another participant are named Send[Entity] and Receive[Entity].

(Rule 4) All management capabilities communicating an entity are provided by a management service called [Entity]TransferService.

(Rule 5) For all management capabilities that are not provided by a service after applying rules (Rule 2) and (Rule 4), a management service is introduced per management activity and named [ManagementActivity]Service.

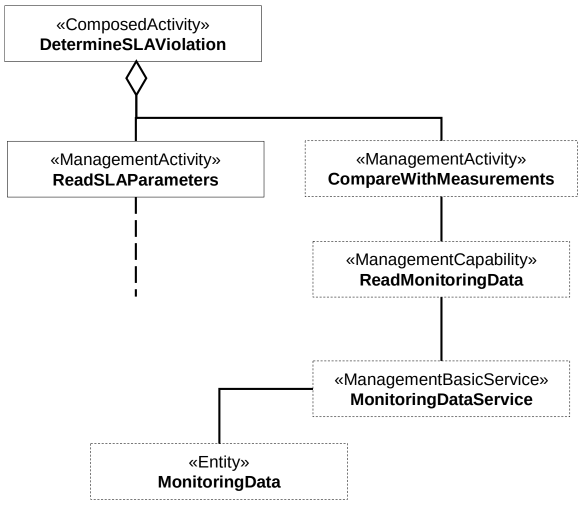

Figure 4. PrioritizeIncident - technically motivated sub activities

Figure 3 illustrates how these rules are applied to the management activity prioritize incident. The two sub-activities are both process motivated, found from the requirements given in ISO20000-1:2005. The management capabilities and services are named according to rules (Rule 1) and (Rule 2).

In Figure 4, one sub-activity is further extended, exploiting the fact that all major management architectures provide methods to read measurement data of managed components. The naming of the management capability and the management service is again done using the rules (Rule 1) and (Rule 2).

D. *Designing management services*

After the domain model was used to identify the conceptual management services needed and the management capabilities they should provide, the services can be modeled using SoaML. This is done by using some more transformation rules.

(Rule 6) Each conceptual management service in the domain model translates to a SoaML Capability of the same name.

(Rule 7) A management service found by applying rule (Rule 2) is given “CRU” (Create, Read, Update) operations named Create[Entity], Read[Entity] and Update[Entity]. A delete operation is intentionally left out because deletion of entities is never done according to ISO20000-1:2005.

(Rule 8) A management service found by applying rule (Rule 4) is given to the operations Receive[Entity] and Send[Entity].

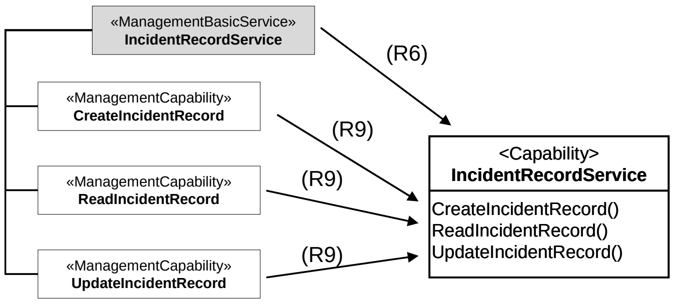

Figure 5. From the domain model to the SoaML Capability

(Rule 9) Operations are added to each SoaML Capability, according to the management capabilities this service should provide, as long as they are not already present after applying rules (R7) and (R8). The operations are given the same names as the management capabilities they should support.

(Rule 10) A SoaML ServiceInterface is created for each management service that exposes the corresponding SoaML Capability.

(Rule 11) A SoaML DataType is created for each Entity with the data fields given by the corresponding entity structure policy.

(Rule 12) The operations Update[Entity] and Send[Entity] are given an input parameter of type [Entity] (the SoaML DataType).

(Rule 13) Operations Create[Entity], Read[Entity] and Receive[Entity] are given an output parameter of type [Entity].

(Rule14) The operations Read[Entity] are given an input parameter of type String, named [Entity]ID. The [Entity]ID is the unique identifier for an entity.

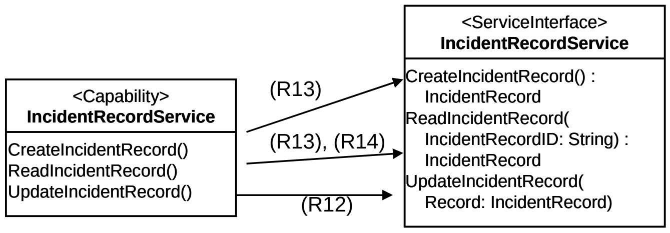

Figure 6. Designing the ServiceInterface

Figure 5 illustrates rules (Rule 7) to (Rule 9) as applied to the IncidentRecordService. In Figure 6, rules (Rule 10) to (Rule 14) are applied in order to model the final ServiceInterface for the IncidentRecordService.

E. *Integrating existing tools*

In the previous sections, the integration of existing management tools was prepared by introducing management activities motivated from common concepts of management architectures. Therefore, to actually integrate an existing tool, all that has to be done is that the appropriate management basic service that exposes the capabilities provided by this tool has to be found.

For example, a tool like Nagios, which focuses on the technical management of components, provides everything needed for the "MonitoringDataService" shown in Figure 4. In order to integrate this tool, development of a webservice adapter is needed, so that the tool exposes its functionality according to the service interface modelled following the rules given in the previous section. Different methods for the development of webservice adapters were suggested before, e.g., in [27], where the authors introduce the concept of mismatched patterns between an existing and a needed service interface. It will be up to the implementing organization to choose the method that best suits their existing tools and requirements.

V. IMPLEMENTATION EXPERIENCE

A discussion of the applicability of our approach includes both a presentation of the achieved results and a generic estimation of the benefit of our method. To address this, in this chapter we briefly present the artifacts that where created along an integration project we currently run at the ATIS, a mid-sized service provider that operates the IT infrastructure in responsible for the faculty of informatics at the Karlsruhe Institute of Technology.

One major goal of this project is to create an integrated management platform that enables both users of the provided IT services and the operators of these services to access relevant management information in one web portal. Furthermore, interfaces to provide management functionality should be created that can be used by external providers connected to the network of the ATIS. During the analysis of the actual situation it became obvious that in order to fulfill these three integrative requirements, a supporting architecture needs to be flexibly adaptable thus architectural elements that are highly reusable had to be engineered. As indicated by some internal examinations, the handling of service disruptions was one of the most urgent use cases that should be implemented at first.

According to ISO20000-1:2005, handling of service disruptions is performed by the Incident Management and Problem Management Process, which in turn is supported by Configuration, Change and Release Management Processes. While the Incident Management deals with restoring disrupted services as fast as possible Problem Management concerns itself with investigation of the root causes leading to recurring service failures. In orderto reduce complexity, we decided to first implement the Incident Management Process, followed by an implementation of Problem Management that can be realized based on the services we identified during the design phase for Incident Management.

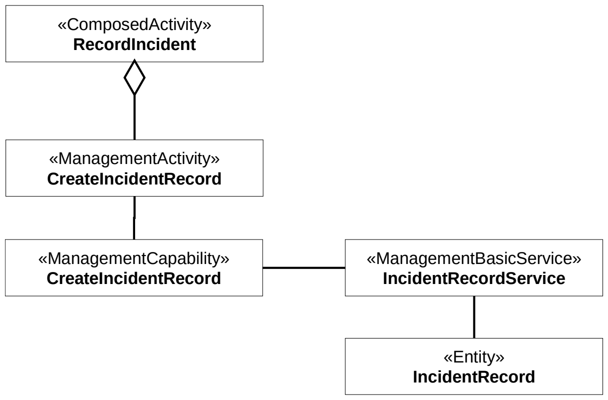

Figure 7. Domain Model for IncidentRecordService

Figure 7 shows an excerpt of the domain model that serve as a starting point for designing management services for the Incident Management Process. As mentioned in Section IV.B, the elements of the domain model can be identified using the definition of Incident Management given in [29].

While the construction of the domain model is fairly straightforward, applying the transformation rules to design the service models currently requires detailed knowledge of the semantics of the modeling elements. Tool support would be highly desirable in this step in order to minimize failures due to semantical misunderstandings. Nevertheless, it turned out to be useful to reflect on the derived service models by both the supporting analyst and the developers alike.

In Figure 8, the results of the transformation applying the rules given in section IV.C and IV.D to the domain model of the IncidentRecordService is given. For instance, applying rule 7 extends the capabilities of the identified operation CreateIncidentRecord with the respective Read and Update operations. As the initial milestone of the development project was rolled out and the additional requirement to implement the Problem Management Process came up, we could reuse the models designed so far and further use service functionality that was already considered during the design phase of the Incident Management services.

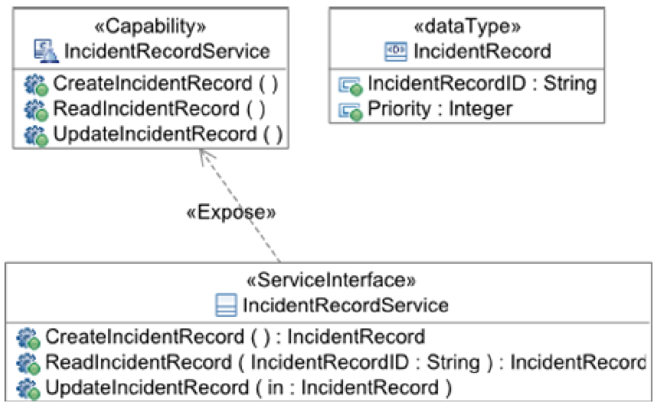

Figure 8. Domain-driven Design of a Management Service for Incident Management

Finally, these designed service interfaces can be implemented using Web Service Definition Language, as shown in Figure 9.

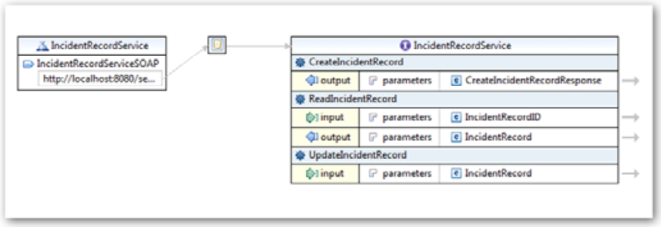

Figure 9. Implementation of the IncidentRecordService using WSDL

In reviewing the lessons learned this small excerpt of the development project shows that application of the method leads to a standardized vocabulary thereby enabling reuse of existing design models when extending systems to support further requirements. The long-term benefit is grounded in the fact that the more the processes will be implemented, the higher the degree of reusability of existing services will be. In order to estimate complexity, we are currently investigating an approach for creating a domain meta-model-based reference model, which includes relations of management functions of different management services for different management processes. This would allow determination of the best starting point for a concrete development project if requirements are clearly given. Figure 9 outlines an early result that served as a basis for our decision to initially implement Problem Management followed by Incident Management.

To sum up the additional overhead of introducing formal domain modeling fairly, we feel that at least basic skills in structural modeling are needed. As the domain meta-model and the related specific instances only make use of classes and their relationships and the fact that the transformation rules were given in natural language, acceptance by the developers involved was surprisingly high. This was because we could show the benefit of formal domain models when extension of given systems was focused. Since the example shows that some of the management basic services needed to implement Problem Management were already identified during analysis of the Incident Management Process, we expect that the implementation of further requirements (Release Management Process, Change Management Process, and Configuration Management Process) will take even less time in terms of c reated design models.

VI. Conclusion and Outlook

In this paper, we motivate the advantages of applying a structured and well-founded development approach in the design of adaptable management services. Since business requirements are constantly changing, the support of management systems for operating IT services has to be able to keep step with this development. Therefore, we promote organized management functionality in terms of loosely coupled management services to enable both a controllable execution of management processes and to adapt to changing requirements.

One of the most critical questions regarding the design of services is to ensure that certain design principles are met. For instance, services are expected to be technology independent, reusable, accessible with defined interfaces and protocols or aligned with business requirements. A major goal of our approach is to support developers of management systems to be supported with some typical engineering instruments enabling evaluation of their management services design against these typical service principles. To reach this goal, we introduced a development approach that is driven by a sound understanding of the domain IT Management. While key concepts of a meta-model for the domain were already introduced in [1], in this paper these key concepts have been revisited in order to integrate an automated verification of derived management services. Furthermore, we extended the development approach in [1] by applying a model-driven approach for designing concrete interfaces for management services. These interfaces can be used to both implement new management services or in scenarios where an integration of existing management tools is necessary, to create integrative adapters to legacy applications. A concrete outcome of our work is a defined set of management services that are needed to execute a typical incident management process. This set of services is aligned with the domain and fulfills several critical service principles, therefore we expect that not only can our conceptual contribution be applied in further scenarios, but also vendors of concrete management tools capture can verify the capabilities of their tools to evaluate the alignment with standard requirements.

Some of the aspects that are presented in this paper need to be discussed further. For instance, a formalized meta-model would allow creating development supporting tools, and simplifying the derivation of services as model-driven techniques could be used. Currently we are exploring approaches to formalizing the presented derivation rules, such as using Object Constraint Language (OCL) [28]. This would be integrated with a formalized meta-model to support a model-driven approach enabling the automated derivation of management services. Furthermore, we are currently considering the integration of existing management tools that would serve as implementation for some management services. For instance, in a typical provider scenario, it is very likely that at least trouble ticket tools exist to coordinate the execution of the incident management process. Creating integrative service adapters in a bottom-up driven way would allow both reuse of existing tools and thecreation of flexible support for workflow support of the management process. We expect that our concept of a combination of domain modeling and rule-based derivation of services can be applied in domains other than IT management.

References

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

[2] V. Machiraju, C. Bartolini, and F. Casati, "Technologies for Business-Driven IT-Management", Proc. Extending Web Services Technologies: the Use of Multi-Agent Approaches“, edited by Cavedon, L., Maamar, Z., Martin, D. and Benatallah, B., Kluwer Academic

[3] G. Aschemann and P. Hasselmeyer, "A Loosely Coupled Federation of Distributed Management Services", Journal of Network and Systems Management, Vol 9, No. 1, 2001, pp. 51-65

[4] N. Anerousis, "An Architecture for Building Scalable, Web-Based Management Services", Journal of Network and System Management, Vol. 7, No 1., 1999, pp. 73-104

[5] C. Xiao, Z. Lv, and S. Zhang, "WS-CHMA: A Composite-Pattern Based Hierarchical WS-Management Architecture", Services - I, 2009 World Conference on (2009) pp. 773 – 780

[6] Z. Lu, Y. Wu, C. Xiao, S. Zhang, and Y. Zhong, "WSDSNM3: A Web Services-based Distributed System and Network Management Middleware Model and Scheme", The 9th International Conference for Young Computer Scientists, ICYCS 2008, 2008, pp. 392-397

[7] Z. Lu, J. Wang, Y. Wu, J. Wu, and Y. Zhong, "MWS-MCS: A Novel Multi-agent-assisted and WS-Management-based Composite Service Management Scheme", IEEE International Conference on Web Services, ICWS 2009, 2009, pp. 1041 - 1042.

[8] Z. Lu, J. Wu, S. Zhang, and Y. Zhong, "Research on WS-Management-based System and Network Resource Management Middleware Model", IEEE International Conference on Web Services, ICWS 2009, 2009, pp. 1051 - 1053.

[9] World Wide Web Consortium (W3C), "Web Services for Management (WS-Management)", Version 1.1.0

[10] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Distributed Management (WSDM) - Management Using Web Services

[11] G. Tamm and R. Zarnekow, "Umsetzung eines ITIL-konformen IT-Service-Support auf der Grundlage von Web-Services", 7. Internationale Tagung Wirtschaftsinformatik 2005 (Bamberg), 2005, pp. 647-666

[12] A. Brown and A. Keller, "A Best Practice Approach for Automating IT Management Processes", Management of Integrated End-to-End Communications and Services, Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium, NOMS 2006, Vancouver, Canada, April 3-7, 2006, pp. 33-44

[13] C. Mayerl, T. Vogel, and S. Abeck, "SOA-based Integration of IT Service Management Applications", Proceedings IEEE International Conference on Web Services 2005, pp. 785-787.

[14] C. Mayerl, T. Tröscher, and S. Abeck "Process-oriented Integration of Applications for a Service-oriented IT Management", The First International Workshop on Business-Driven IT Management, 2006, pp. 29-36

[15] J. Sauve, A. Moura, M. Sampaio, J. Jornada, and E. Radziuk, "An Introductory Overview and Survey of Business-Driven IT Management", BDIM '06. The First IEEE/IFIP International Workshop on Business-Driven IT Management, 2006, pp. 1-10.

[16] Office of Government Commerce (OCG): IT Infrastructure Library (ITIL) – Service Support (ISBN 0113300158), 2000; Service Delivery (ISBN 0113300174), 2001; Planning to Implement Service Management (ISBN 0113308779), 2002; Application Management (ISBN 0113308663), 2002.

[17] V. Tosic, "The 5 C Challenges of Business-Driven IT Management and the 5 A Approaches to Addressing Them", Business-Driven IT Management, 2006. BDIM'06. The First IEEE/IFIP International Workshop on Buisness-Driven IT-Management, 2006, pp. 11-18.

[18] T. Erl: SOA Principles of Service Design. Prentice Hall Service-Oriented Computing Series, 2008.

[19] S. Cohen, "Ontology and Taxonomy of Services in a Service-Oriented Architecture", The Architecture Journal, Volume 11, 2007

[20] M. Gebhart and S. Abeck, “Rule-based service modeling”, Fourth International Conference on Software Engineering Advances, 2009. ICSEA '09. , 2009, pp. 271 - 276

[21] 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)

[22] Object Management Group, “Service oriented architecture Modeling Language”, http://www.omg.org/spec/SoaML/1.0/Beta2/, 04/2009 (last visited 08/10/2010)

[23] V .Machiraju, C. Bartolini, and F. Casati "Technologies for Business-Driven IT Management", In: Extending Web Services Technologies: The Use of Multi-Agent Approaches (Multiagent Systems, Artificial Societies and Simulated Organizations), Springer, 2005, pp. 1-27.

[24] M. Brenner, "Classifying ITIL Processes – A Taxonomy under Tool Support Aspects", Proceedings of First IEEE/IFIP International Workshop on Business–Driven IT Management (BDIM 06), pp. 19–28, April, 2006.

[25] International Standards Organization, “Open Systems Interconnection – Basic Reference Model – Part 4: Management framework”, ISO/IEC 7498-4, 1998

[26] Distributed Management Task Force: Web-Based Enterprise Management (WBEM), http://www.dmtf.org/standards/wbem/ (last visited 08/10/2010)

[27] B. Boualem, F. Casati, D. Grigori, M. Nezhad, and F. Toumani, “Developing Adapters for Web Services Integration”, In Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE), Porto,Portugal CAiSE, 2003, pp. 415-429

[28] OMG, “Object constraint language”, Version 2.0, 2006.

[29] International Standards Organization, "Information technology — Service management —Part 1:Specification, ISO/IEC 20000-1, 2005

[30] The Workflow Management Coalition, "Workflow Management Coalition Terminology & Glossary", 1999