Enterprise methods management using semantic web technologies

Peter Rosina, Bernhard Bauer

Angaben zur Veröffentlichung / Publication details:

Rosina, Peter, and Bernhard Bauer. 2015. "Enterprise methods management using semantic web technologies." In *Proceedings of the Fifth International Symposium on Business Modeling and Software Design, July 6-8, 2015, in Milan, Italy*, edited by Boris Shishkov, 46–55. Setúbal: SciTePress. https://doi.org/10.5220/0005885400460055.

Enterprise Methods Management Using Semantic Web Technologies

Peter Rosina and Bernhard Bauer
Software Methodologies for Distributed Systems, Institute of Computer Science
University of Augsburg, Universitätsstr. 6a, 86135 Augsburg, Germany
firstname.lastname@ds-lab.org

Keywords: Methods, modeling, ontologies, semantic web, enterprise architecture.

Abstract: Due to today's product complexity and variety and a shortening of development cycles, for instance, in the automotive domain, a conventional knowledge representation and management of the various design methods is not reasonable anymore. By acquiring the relevant domain and business knowledge from IT applications, documents and experts and creating ontologies, business rules and queries thereof, domain experts can manage this knowledge independently and thus react faster to the ever-changing development process.
Using Semantic Web technologies, we created a methods ontology that enables domain experts to analyze and compare method meta knowledge, e.g., about physical and virtual CAx methods. Furthermore, this particular domain and business knowledge features interdependencies with the remaining enterprise knowledge, including business processes, organizational aspects and the IT architecture. Therefore, we show concepts for the integration of this method meta knowledge into an enterprise architecture.

1 INTRODUCTION

The task of R&D divisions in the automotive domain comprises forming ideas for new products, i.e., the product planning, and the design in different stages until it is ready for the start of production in the production division. Naturally, a multitude of different disciplines and departments, for instance, physical crash tests and virtual simulations, have to collaborate, cooperate and interact during this process which entails the application of a large number of various technologies and working methods (*cf.* section 3), for instance, Computer Aided x (CAx) methods. CAx comprises the Computer Aided Design (CAD) that provides approaches for the virtual design and verification of products and their geometry. Further involved disciplines are the Computer Aided Engineering (CAE) and Testing (CAT); the former provides methodologies for simulating the behavior of a car and its functions, e.g., Finite Element Analysis (FEA) for crash simulation; the latter for performing physical tests, e.g., vehicle management, job, and testing control, and test result analysis. The involved tools and methods range from mechanical test beds, through Hardware in the Loop (HiL) to Digital Mock-Ups and other pure software applications.

Our goal is to create an integrated model that depicts this domain of methods, technologies and tools in order to support stakeholders in their daily work. Nowadays, a method selection for a defined purpose, an analysis of the entirety of methods in a company and the method monitoring are mainly manual tasks, ever and anon assisted by basic IT documents, like spreadsheets. Formalizing this knowledge allow the involved roles to analyze methods along the product development process, for instance, for finding virtual replacements for physical tests. This meta model and the resulting ontology are introduced and explained in section 4.

We realized our meta model, resulting ontology and a prototypical application, following the ONTORULE approach (de Sainte Marie et al., 2011). Using Semantic Web technologies (*cf.* section 2.1) bears many advantages: on the one hand, it enables a clear separation of domain and business knowledge, that can be extended and modified by domain and business experts. On the other hand, this knowledge can be implemented and managed independently from the

Rosina P. and Bauer B.
Enterprise Methods Management Using Semantic Web Technologies.
DOI: 10.5220/0005885400460055
In *Proceedings of the Fifth International Symposium on Business Modeling and Software Design (BMSD 2015)*, pages 46-55
ISBN: 978-989-758-111-3

applications that process it.

Finally, in order to benefit mutually from the additional knowledge available in an enterprise, we illustrate how to combine our method meta model with an enterprise architecture (EA) meta model in section 5. The basics about EA are introduced earlier in section 2.2. In section 6, we shortly present a prototypical demonstrator showcasing the method meta ontology and its application. Furthermore, we discuss the opportunities that emerge by combining method and EA meta models. The paper is then summarized in the final section 7.

2 BASICS

2.1 Semantic Web Technologies

Bringing the Semantic Web to fruition is an collaborative endeavor of various research groups, industrial partners and organizations pursuing the common goal to enrich the WWW with semantic data, thus transforming the web from an un- or semi-structured document based technology to a knowledge-centered one. (Berners-Lee et al., 2001) developed the vision of the Semantic Web in order to make the web's meaning machine-readable and -interpretable and hence integrate heterogeneous data and infer new information from existing knowledge bases (KBs) automatically.

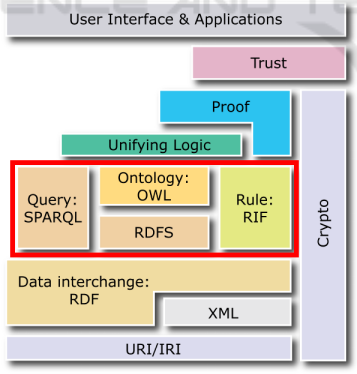

Figure 1: Semantic Web layercake (after (W3C, 2007)).

The prominent Semantic Web Stack in Figure 1 depicts the architecture of the Semantic Web, which is still in development and continuously improved. The lower layers provide the technological basement for the subsequent upper layers. The highlighted layers' containing formats, i.e., RDFS, OWL (W3C OWL Working Group, 2012), RIF (Kifer and Boley, 2013) and SPARQL (Harris and Seaborne, 2013), serve as our technological foundation. This paper represents an excerpt of our ongoing work, though. Therefore, we focus on modeling ontologies with the knowledge representation language OWL. OWL ontologies separate a terminological box (TBox), representing the ontological concepts, and the modeled individuals in the assertional box (ABox). Moreover, we do not necessarily apply these technologies for the WWW, but for the enhancement and realization of enterprise domain models and their applications.

2.2 Enterprise Architecture Management

Many enterprises' IT landscapes have reached a degree of complexity that is only hard to understand and manage (Hanschke, 2013). Additionally, the IT needs to be aligned to business services and processes for an optimal and efficient support. For this purpose, many organizations have established an EA Management (EAM), motivated by the circumstance, that business is changing faster and faster due to shorter development cycles, adaptation and reorientation of business models and an overall need for improving the business and IT alignment.

EAM helps companies to reduce costs for maintenance and developments when confronted with an increasing number of systems by analyzing "areas of common activity within or between organizations, where information and other resources are exchanged to guide future states from an integrated viewpoint of strategy, business and technology" (EABOK Consortium, 2015).

Success factors are a goal-oriented adaption towards changing market and boundary conditions, the detection of redundancies and the definition, determination and an ongoing assessment of the various EAM Key Performance Indicators (KPIs) (Auer et al., 2011). "Above all, [EA] recognizes that the information assets of an enterprise are always in flux, and that this flux is the steady state" (Wood, 2010).

An EA's centerpiece is its meta model. It represents the diverse required aspects of a company's IT and business structures. These are, for instance, elements such as processes, services, organizational structures, data, applications and technologies. They are connected and organized via various relations and clusters which form the EA. The IT architecture can be further subdivided into architectures covering software and technology.

Every company is unique, with diverging requirements, goals and focus areas. For this reason, a suitable, customized meta model is more important when introducing an EAM in a company than a tool which comes along with an inflexible standard meta model.

An EA Framework (EAF) describes a methodol-

ogy for developing an EA and its use during operation. It points out the relevant aspects and focuses that an enterprise should consider when creating information systems. Some examples for EAFs are *The Open Group Architecture Framework (TOGAF)* (The Open Group, 2011), *ISO 19439:2006* or the *US Federal EAF*, to name just a few (Matthes, 2011).

TOGAF, together with ArchiMate (The Open Group, 2013), serves as the foundation for our EA meta model. It provides a methodology for the design, planning, development, implementation and maintenance of an EA (The Open Group, 2011). It does not feature a specific model the companies can use, but offers meta models and guidelines that should be applied as seen fit and as required.

TOGAF partitions an EA into four main architecture tiers: the *Business*, *Data*, *Application* and *Technology architecture*. The *Data* and *Application Architecture* are part of the *Information Systems Architecture*. In this paper, we mainly focus on the business level and its relations to the information systems, for instance by dealing with business strategy, processes, organizational structure and business capabilities and the applications relevant for the execution of the business processes.

The meta model along with suitable tools and an EAF allows the enterprise to document and monitor the business and IT architecture in a holistic way which enables the architects and managers to react faster to occurring changes in an agile enterprise. Furthermore, modeling the business and IT architecture leads to an increased transparency which in return drives the use of a common vocabulary in the company and hence reduces misunderstandings. Other benefits are the advancement of standardization, an improvement and assurance of quality, a reduction of IT costs and consequently an improved coping with risks.

Generally speaking, the above listed advantages should provide a fillip for any organization. However, introducing an EAM is especially worthwhile for big enterprises. Such an enterprise can even be an “extended enterprise”, that “nowadays frequently includes partners, suppliers and customers”, as well as internal business units (The Open Group, 2011). On the one hand, small organizations shun the undertaking, because it is wedded to a lot of effort – time- and resource-wise. On the other hand, a big enterprise, with an historically evolved IT landscape and complex business processes, will see the most benefits from such an endeavor.

3 RELATED WORK

Our model has been mostly influenced by the domain of CAx methods, in particular, our primary goal was to model and analyze CAD, CAE and CAT methods. However, we soon discovered, that a method meta model covering not only CAx, but a much bigger selection of working and design methods is of even greater value.

Depending on the department or company, even of the same domain, e.g., automotive engineering, a method can be synonymous to a *tool*, a *process*, a kind of technique for solving or analyzing problems or a combination thereof. In software engineering, the term is usually associated with a *procedure*, i.e., a segment of a framework, of a software development methodology, e.g., the waterfall model or an agile development framework.

This ambiguity impedes the communication between people with a different background and leads to unnecessary misunderstandings and coordination phases. Thus, a standardization of the semantics would be worthwhile.

Methods have been used for decades in the domain of Design Theory and Methodology – an approach for the methodical development of products by using “effective methods to support particular development steps and [guide users] to efficiently solve development tasks” (Birkhofer, 2011). However, according to (Pahl et al., 2007), the industry only reluctantly adapts design methodological models and methods.

By definition, a method is a systematic *procedure* for the *attainment* of something (Oxford Dictionaries, 2015) (Merriam-Webster.com, 2015), for instance, for the attainment of [scientific] insights or practical results (Duden, 2015).

Because our running example originates from the automotive domain, we also incorporate domain specific method definitions and are inspired by existing method frameworks, e.g., from systems engineering and method development (*cf.* (Weigt, 2008)).

A method consists of one or many *procedures* that are interconnected by logical rules (*alternative*, *predecessor*, *successor*, etc.) (Hesse et al., 1992) (Chroust, 1992). Methods are prescriptive which means they are perceived as some kind of instruction or plan (Lindemann, 2009) and they cannot be applied universally, but *every method has a particular stage in the development process* where its execution offers the best outcome (*cf.* (Meerkamm, 2011)). More precise, methods are used in development processes for supporting a systematic and *aimed execution of tasks* (Lindemann, 2009). Additionally, plans

and methods must be adapted to the specific situation they are used in which means that a method can produce entirely different results and qualities depending on its environment. Among others, this includes the user itself. They have different knowledge, experiences, skills, tendencies and their form on the day is variable as well. Furthermore, the quality *depends on the tools* that are used to execute a method, the type of *business process* a method supports and of course the *quality of its input parameters*.

According to (Müller, 1990), methodical support should be coordinated with its respective boundary conditions. In particular, it should be differentiated between the boundary conditions of the *organizational and operational layers* (Weigt, 2008). Boundary conditions of the organizational kind include *role descriptions*, the mapping to *business processes*, *organizational structures* etc. The operational layer comprises information about the *tools* used to conduct the method, its *qualities*, *time consumption* and *costs*.

(Cronholm and Ågerfalk, 1999) define the relations between the different method concepts in even more detail by defining the relations between them: methods can be vertically linked, thus creating *method chains* – the outcome of the previous method is the input for the next method. A *method alliance* on the other hand, is a horizontal composition of methods, i.e., a family of similar, alternative methods. Another important concept is the *perspective*, i.e., how methods are perceived and what level of detail and focus is presented to the user.

Another significant aspect for the definition of methods, more precise for the framework of methods (Cronholm and Ågerfalk, 1999), is the *methods' structure*. According to (Lindemann, 2009), it is not a simple task to clearly classify methods and put them in some hierarchical structure. A kind of network or graph, e.g., an ontology, is best suited for this task, because single methods and their partial procedures can be applied in other methods as modules as well which "supports a flexible selection, adaption and combination of methods" (Lindemann, 2009).

A method typically consists of the five bullet points listed below (Lindemann, 2009):

Purpose / Goal: A task in the development process which is supported by the method.

Situation: The scope, problem descriptions and boundary conditions the method is usually appropriate for.

Effect: Effects and side effects, that are attained by executing the method, i.e., the method's output.

Procedure: The performed steps when executing the method.

Tools: Form sheets, check lists, software, test beds, etc.

Braun and Lindemann have analyzed the selection, adaption and application of product development methods for the impersonal transfer of method know-how, which resulted in the Munich Model of Methods (MMM) (Braun and Lindemann, 2003). It acts as the foundation for a method model which consists of method building blocks that are linked by method attributes. Furthermore, the model supports the implementation of *superior tasks* and *resources* and *support* (*cf.* Figure 2) can be related to a procedure.

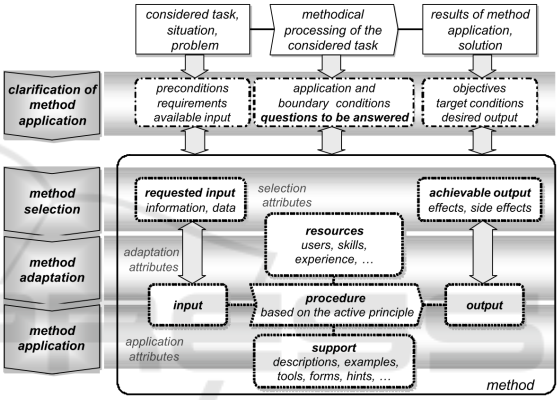

Figure 2: The *Munich Model of Methods* (Braun and Lindemann, 2003; Lindemann, 2009).

The MMM comprises various steps for applying a method.

The diagram depicted in Figure 2 is horizontally split into four phases. It juxtapositions method attributes and the method implementation during the *process action*. The first lane deals with the clarification of the method's use case scenario. The following lanes represent the method selection, its customization and its application.

Vertically, the model is divided into different process building blocks, i.e., the requirements phase, boundary conditions and concerns concerning the method application and of course the method's *goal* and *output* in the third vertical lane (Braun, 2005).

The single phases are supported by various *tools*.

The majority of the depicted blocks are also covered by our method ontology, because the entities concerning a method definition, like *input*, *output*, *goal*, *procedure* etc., necessarily have to be implemented in a model when dealing with design methods.

4 METHOD META MODEL

Our method meta model allows to semantically express the relations of tools and methods in combination with their tasks and processes on an abstract layer but also offers the possibility to describe concrete methods and their boundary conditions, e.g., the available time or budget.

Furthermore, our model allows to annotate the quality of their interactions, based on a maturity scale. To achieve a common understanding, we created a method meta model that has been implemented with OWL later on, by working closely together with industrial partners and regarding state-of-the-art definitions and models. Further necessary knowledge that has been derived from our gathered requirements, i.e., to answer requested information, has been implemented in the form of queries and rules. These are not presented in this paper, though.

4.1 Method Ontology

The resulting method ontology's core structure is depicted in Figure 3. We use the attribute "core", because this ontology can and should be extended with further ontologies. The ellipses in the figure represent ontology classes and the directed edges stand for properties, whereas their beginning is the property's domain and the arrowhead represents its range. Multiple properties between the same classes are consolidated into a single edge for a better overview; they are separated by commas between the edges' labels, though.

When talking about a method's maturity, quality or other concerns, we have to take into account, that a method can be applied at several points in a company's processes. Usually, the quality of data or resources rises over time, when the product itself is becoming more and more ready for series production. Assuming that we have a method A in our method KB which can be applied at two different points of time in the product development process, it is conceivable that this method is conducted using input resources of a differing quality. Vice versa, the method's application is resulting in similar, but of differing quality, resources which have to be available at a specific point of time in the process, e.g., a milestone. Nonetheless, it is still the same method, supported by the same tools and performed by similar procedures. Independently, a method can also be applied with the same, or at least the same quality and kind of, resources multiple times in a process. In order to distinguish between these (concrete) instances of a method and the general (abstract) method they have been derived from, we have introduced the concepts `Concrete Method` and `Abstract Method`. An abstract method M_a is a concept that defines a class of process-independent methods. A concrete method M_c is a concept that defines a class of methods, linked to a process action and derived from an abstract method M_a.

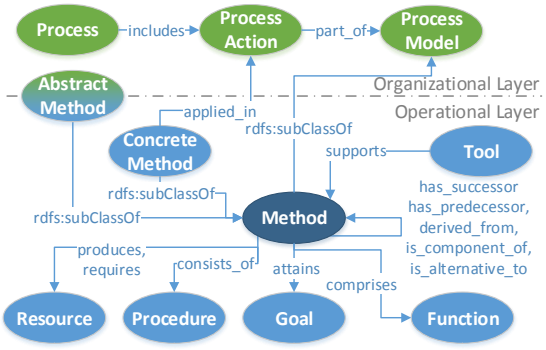

Figure 3: Structure of the core method ontology.

In Figure 3, both, the `Abstract Method` and the `Concrete Method`, are specializations of the concept `Method`. The `Method` is a specialized type of a `Process model`. Methods offer suggestions for specific tasks' sequences and the fashion on how these tasks are to be conducted (Lindemann, 2009). Following the example of (Eisenbarth, 2013), a `Process Action` is equivalent to a task and hence, a `Concrete Method` is `applied_in` a specific task that is again related to a `Process`. Because both method types inherit the object properties of `Method`, we can model interrelationships between `Concrete Methods` and respectively `Abstract Methods`, and can also state that some arbitrary `Concrete Method` is `derived_from` a specific `Abstract Method`. The remaining object properties allow us to model classes of methods such as method chains and alliances (`has_successor`, `has_predecessor`, `is_alternative_to`).

The three concepts at the bottom right of Figure 3, namely `Procedure`, `Goal` and `Function`, are used to formally describe the methods' contexts, resp. its comprised procedures (*cf.* section 3).

An instance of the class `Procedure` can either describe this method's procedure in textual form with the given RDFS properties (*rdfs:comment*, *rdfs:seeAlso*, *rdfs:label*, ...) or can be linked to more complex constructs, like a class `Document` (not part of the figure) which again may link to a document in the file system. Otherwise, if a method makes use of another method, object properties, like `is_component_of` shall be used.

Taking into account, that a method is al-

ways purposeful and therefore always focused on a solution of a problem or task (Lindemann, 2009), a `Method attains` the concept `Goal`. The method's `Goal` is the class that can describe the method's contribution to the enterprise's strategy or overall value creation. For example, we can create the classes `ProductAssurance` or `VehiclePropertyAssurance` as a `Goal`'s subclasses. When a specific method assures an arbitrary business product, the respective instance can be linked to the `Product` of a product ontology.

Usually, a `Method` is `supported` by a number of `Tools` that are to make its application more effective and efficient (Lindemann, 2009). The term and therefore the respective class `Tool` covers a wide range of different auxiliaries or assistive equipment. In the domain of CAx, this can be all kind of testing equipment, physical implements, simulation or modeling software, but also arbitrary business software, statistical analyses or something totally different. Besides, the term also comprises simple assistive things, like forms, checklists etc.

4.2 Extending the Method Ontology

Up to now, the introduced method model allows to observe and compare methods based on their semantic context information, assumed the appropriate SPARQL queries have been implemented, for instance, the application of methods and tools during a specific process can be queried. Additionally, we further extend the method ontology with other ontologies, covering metrics, resources, descriptions, enterprise vocabulary etc., in order to model and be able to analyze a company's method landscape in a holistic way.

Due to limited space, we can only introduce the overall architecture and a summarized description of the created ontologies, though.

One of the big advantages of using Semantic Web technologies is having this vast amount of publicly available community ontologies, for instance, FOAF (Brickley and Miller, 2014) or SKOS (Isaac and Summers, 2009), that can be used as extensions, geared to the company's needs. The architecture presented in Figure 4 illustrates how our core method ontology is integrated as an upper ontology.

The remaining ontologies, e.g., a process, resource and metrics ontology, are mapped to the core ontology.

Independent from the chosen mapping paradigm between the ontologies, a company wants to model its own KB, covering their use cases. This domain ontology is depicted in Figure 4 as *Method Ontology X*.

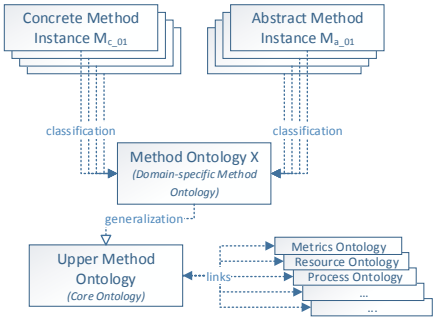

Figure 4: The method ontology's architecture.

The domain ontology specializes the concepts of the upper ontologies with domain specific ones and provides the TBox for our specialized assertions (ABox), i.e., *Concrete Method Instances* M_{c_u} and *Abstract Method Instances* M_{a_v}.

For instance, it can be used to provide a necessary taxonomy for the domain of CAx methods by introducing new concepts, e.g., `CAE Method`, `CAD Method` and `CAT Method` or a more general method, like *dc:MethodOfInstruction* from Dublin Core. Along with new kinds of methods, apposite, more specialized metrics or other arbitrary parameters can be introduced in the same way.

The knowledge for creating an own domain-specific ontology can either be acquired from existing company sources, like documents, or from experience, by interviewing experts as described in the ONTORULE methodology (de Sainte Marie et al., 2011).

In order to compare methods based on their KPIs, for instance, the method's cost, input and output quality, maturity or duration, we developed a metrics ontology, covering the required quality attributes. Next to selection criteria, the metrics can act as indicator for strengths and weaknesses in the company's method framework and allows to detect gaps, e.g., processes that are scarcely supported by methods. This information is valuable for the strategic method development. The model can also be extended with arbitrary execution or evolution quality attributes (the so called "ilities"), like usability, reliability, manageability etc. Which kind of non-functional requirements to choose is facultative and up to the company's knowledge and methods engineers.

In accordance with the distinction between concrete and abstract methods (section 4.1), we also distinguish between `Concrete Resources` and `Abstract Resources` in the developed resource ontology. This model covers the required inputs and produced outputs of our methods. When modeling an `Abstract Method`, the knowledge engineer can define the type of resources that are required or pro-

duced by this method, for instance, a general placeholder parameter for an abstract CAD model name. But not until the `Concrete Method` instance is modeled and assigned to a process action, the `Concrete Resources` can be named, for example, a particular drawing. Following current best practices, like the example of the W3C Product Modelling Incubator Group (W3PM, 2009), our resource ontology, is influenced by a standard product model, namely STEP (AP 214/242) (ISO, 2014). We decided not to model a product's inner structure, like PDM systems do, but to confine ourselves with more abstract concepts, because our method ontology's purpose is still to model only the meta level of the method landscape, i.e., analogous to the method concept, we treat the products as a black box.

Furthermore, we have used SKOS to manage the heterogeneity of the enterprise's vocabulary, i.e., methods and policies, like regulations (Omrane et al., 2011). This way, multiple labels can be annotated to all the ontology's entities which makes them generally intelligible, e.g., by enabling a multilingual use or making the semantics more comprehensible for people with differing professional backgrounds.

5 ENTERPRISE INTEGRATION

The main objective of this integration has been a combination of method knowledge with an EAM in a product development division. This way, the input of strategic, business and IT decisions on methods and vice versa can be easily inferred, while the knowledge can be maintained independently.

Integrating our method meta model into an EAF bears many advantages. First of all, the combined meta models allow stakeholders to perform novel kinds of analyses, like impact analyses or discovering business and IT relations. For example, the concern "Which system/application supports which methods?" or the responsibility of the modeled actors and roles can be identified. Furthermore, a company benefits from such a mapping approach through a concerted and defined meaning of the modeled concepts and vocabulary. OWL, especially the extension SKOS, supports the use of various labels, hence different vocabularies can be attached to the concepts, if required.

As a prerequisite for the integration of our method meta model into an EA meta model, we first need to obtain a formalized model. We have decided to use TOGAF and ArchiMate as a foundation for our EA model because of their popularity and maturity. However, the Open Group does not provide a formalized model of their framework but considers TOGAF as an approach that should be further refined and implemented. Nevertheless, using ontologies and other Semantic Web technologies for the realization of EAs has been done for years now (Ortmann et al., 2014).

As a consequence and inspired by the above mentioned EA frameworks and meta models, we also created our own EA ontology. Since our method meta ontology has been realized using OWL, we also applied this language when designing our EA ontology, depicted in Figure 5.

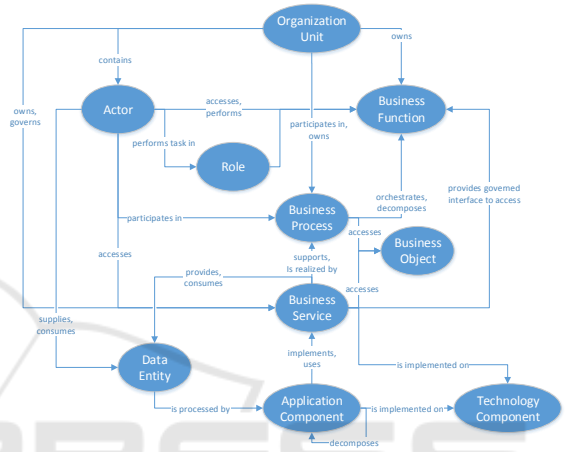

Figure 5: Enterprise Architecture ontology based on TOGAF and ArchiMate meta models.

An important requirement for this EA ontology has been a suitable coverage of the concepts known from our core method meta ontology, together with the extended method ontologies. Thereby, we want to make use of newly generated relations, for instance, from methods to business objects or capabilities. The TOGAF Core Content Metamodel, combined with the Archimate Design approach fulfill this requirement and can be extended when specialized concepts are needed as illustrated in Figure 6. We use the same generic concepts (OWL meta model) and extend the EA concepts with a more specific domain meta model.

After selecting the EA model, we need to establish mappings from our method ontologies' concepts and entities to the already existing elements in the EA.

Technically, we could pursue different mapping techniques for combining our ontologies: we could use the other ontology's concept URIs directly, which is the standard way in Semantic Web. For example, by importing the ontology into ours, we can use *ea:BusinessProcess* as a replacement for our *method:Process* in the method ontology. However, a fundamental idea behind our integration is the retained separation of both KBs, because each domain

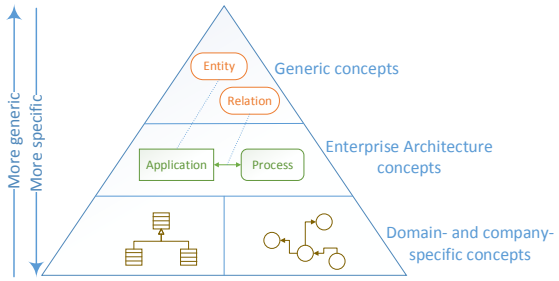

Figure 6: Meta models at different specificity levels (after (The Open Group, 2013)).

– the method knowledge and the EA knowledge – is the particular responsibility of a dedicated organizational unit and hence, changes in the other ontology can be opaque. An alternative option would be to use an upper ontology, like UMBEL (Giasson and Bergman, 2015), which is certainly a reasonable choice when combining lots of domains. We do not need such an explosion of our domain for the scenario at hand, though. The technique we have chosen to map the TBoxes is the use of an own mapping ontology for combining the various concepts, for instance, by using *owl:equivalentClass* or *rdfs:subClassOf*. This option can be implemented very fast, the mappings are traceable in one ontology, and it offers a good overview for a manageable amount of concepts, which is sufficient for the prototypical introduction presented in this paper.

Furthermore, next to matching the ontologies' TBoxes, we make statements about the individuals in the ABoxes that represent our model. The respective individuals, e.g., the modeled processes, are usually matched using OWL notation, as in *owl:sameIndividual*. However, it is often the case, that two individuals are only nearly exactly the same. Therefore, the use of a 'Similarity Ontology', along with object properties like *so:identical*, is a good idea. It allows to express different levels of identity (Halpin et al., 2010).

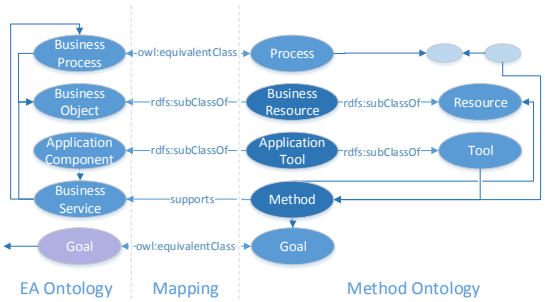

Figure 7: Mapping of EA and method ontology.

When comparing our core method ontology's with our EA ontology's concept names, we encounter some obvious similarities. An equal or similar name, however, does not automatically infer synonymous semantics. The considered key concept `Process` from our method ontology and the concept `Business Process` from the EA ontology are identical, though, especially when regarding the TOGAF "Process Modeling Extension". The conceptual mapping between both ontologies is depicted in Figure 7. We state that both concepts are equal even though the different processes can vary in their granularity.

Our intentions for the methods' input and output `Resources` are semantically covered by the concept `Business Object`, known from the ArchiMate Business Layer Metamodel. This concept "represents a business entity (e.g. an invoice) that is used during the execution of a business process" (Buckl et al., 2008). Processes can perform all kind of create, read, update and delete (CRUD) operations on a business object, which either represents a virtual or physical object (or both) (The Open Group, 2013). This applies to our methods' relations and the corresponding resources, as well. However, only a subset of the modeled EA business objects are pertinent for the analysis of our meta method ontology and vice versa. Therefore, both concepts share the common subclass `Business Resource` (*cf.* Figure 7). This subclass is modeled in a domain ontology, importing the upper method ontology.

The same reasoning applies to the concepts `Tool` and `Application Component`. A lot of business software is of no interest for the method domain and tools of the method domain can also include physical devices and implements. Consequently, we introduce the concept `Application Tool`. The various individuals in the ontologies' ABoxes are then mapped to their counterpart using properties that express the appropriate similarity.

Additionally, `Methods` represent a link between `Business Services`, process actions and tools. They express, how and when a business service can be applied to a specific process.

The final depicted mapping between both ontologies deals with the motivation extension, since methods as well as stakeholders in EA intend to achieve a `Goal`.

The remaining concepts of the core meta method model feature no counterpart in the EA ontology, even though both models feature a concept named `Function`. A function in TOGAF "delivers business capabilities closely aligned to an organization [...]. Also referred to as 'business function'" (The Open Group, 2011), whereas a function in our meta method

model represents an appropriated behavior of a technical system (Weigt, 2008).

6 EVALUATION

The combination of an enterprise architecture with our method ontologies and the respective domain models enable methods engineers, enterprise architects, domain experts and other stakeholders to conduct novel kind of analyses. For example, we can compare and hence select the most suitable methods, based on the modeled KPIs, at specific process phases in the product development. Furthermore, a purposeful development and shutdown of methods is made possible by performing analyses based on the baseline and target architectures.

The new concepts and contributions, namely the use of standard Semantic Web languages, the EA integration, the evolved method meta model, along with the extending ontologies and the corresponding queries and rules, have been modeled, formalized and executed with a chosen set of test scenarios that showcase the proof of concept.

An earlier, specialized version of the introduced method meta model, together with a prototypical implementation as seen in Figure 8, has been developed and published during the FP7 ONTORULE project (Rosina and Kiss, 2011). This demonstrator proofs the feasibility and illustrates the benefits of our approach. However, it has been implemented using an alternative knowledge representation and rule language (ObjectLogic), covering a tailored meta model that realizes a particular automotive use case.

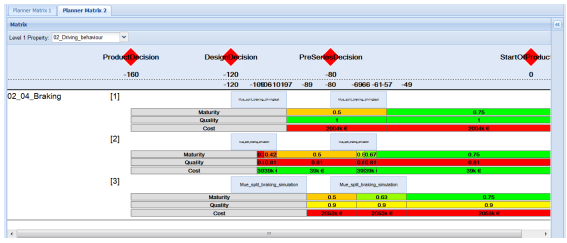

Figure 8: Extract of the demonstrator showing various possible method applications (Rosina and Kiss, 2011).

7 CONCLUSION

The developed ontologies presented in this paper depict the domain of design methods, including their context information, for instance, references to the processes they are used in, quality attributes and the methods' in- and outputs. This allows us to predicate statements on expected qualities, costs, time consumption or other KPIs which is a strong motive for domain experts when selecting an appropriate method best suited for the task at hand. They have been realized using Semantic Web standards, however, we omitted almost the entire technical part in this paper, concentrating on the conceptual models. A former technical realization is referenced in the evaluation section, though.

Another novelty is the integration into an EA ontology, which bears many mutual benefits. It enables the analysis of relationships on a strategic level, fostering a targeted method development, on the business level, e.g., between roles, processes and methods, and also on a business to IT level, for instance, by providing an overview of the applied method software tools.

However, the application, documentation, the execution and the maintenance of a formalized method management is expensive, can be complex and is wedded to effort, because it is mainly a manual procedure.

REFERENCES

Auer, G., Basten, D., Berneaus, M., Däberitz, D., Freitag, A., Haas, H., Kröber, G., Schmidtmann, V., Schweikert, R., Stettiner, E., Thielscher, J., Triebel, T., Weber, M., and Weisbecker, A. (2011). *Enterprise Architecture Management neue Disziplin für die ganzheitliche Unternehmensentwicklung*. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V., Berlin, Germany.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

Birkhofer, H., editor (2011). *The Future of Design Methodology*. Springer-Verlag London Limited.

Braun, T. (2005). *Methodische Unterstützung der strategischen Produktplanung in einem mittelständisch geprägten Umfeld*. PhD thesis, TU Munich, Munich, Germany.

Braun, T. and Lindemann, U. (2003). Supporting the selection, adaptation and application of methods in product development. In *International Conference on Engineering Design, ICED'03*. Design Society.

Brickley, D. and Miller, L. (2014). FOAF Vocabulary Specification. `http://xmlns.com/foaf/spec/`. Retrieved 2015-04-20.

Buckl, S., Ernst, A., Lankes, J., and Matthes, F. (2008). Enterprise Architecture Management Pattern Catalog. Technical report, Software Engineering for Business Information Systems (sebis), Munich, Germany.

Chroust, G. (1992). *Modelle der Software-Entwicklung - Aufbau und Interpretation von Vorgehensmodellen*. Oldenbourg-Verlag, Munich, Germany.

Cronholm, S. and Ågerfalk, P. (1999). On the Concept of Method in Information Systems Development. In Käkölä, T., editor, *22nd Information Systems Research In Scandinavia (IRIS 22)*, Keuruu, Finland.

de Sainte Marie, C., Escudero, M. I., and Rosina, P. (2011). The ONTORULE Project : Where Ontology Meets Business Rules. In *Web Reasoning and Rule Systems*, volume 6902, pages 24–29, Galway, Ireland.

Duden (2015). 'methode'. http://www.duden.de/rechtschreibung/Methode. Retrieved 2015-04-20.

EABOK Consortium (2015). Enterprise Architecture Body of Knowledge. http://www2.mitre.org/public/eabok/index.html. Retrieved 2015-04-20.

Eisenbarth, T. (2013). *Semantic Process Models - Transformation, Adaption, Resource Consideration*. PhD thesis, University of Augsburg, Augsburg, Germany.

Giasson, F. and Bergman, M. (2015). Upper Mapping and Binding Exchange Layer (UMBEL) Specification. http://techwiki.umbel.org/index.php/UMBEL_Specification. Retrieved 2015-04-20.

Halpin, H., Hayes, P. J., McCusker, J. P., McGuinness, D. L., and Thompson, H. S. (2010). When owl:sameAs isn't the Same: An Analysis of Identity in Linked Data. In *The Semantic WebISWC 2010*, pages 305–320.

Hanschke, I. (2013). *Strategisches Management der IT-Landschaft*. Carl Hanser Verlag München, Munich, Germany.

Harris, S. and Seaborne, A. (2013). SPARQL 1.1 Query Language. http://www.w3.org/TR/sparql11-query/. Retrieved 2015-04-20.

Hesse, W., Merbeth, G., and Frölich, R. (1992). Software-Entwicklung - Vorgehensmodelle, Projektführung und Produktverwaltung. In *Handbuch der Informatik*, volume 5.3. Oldenbourg-Verlag, Munich, Germany.

Isaac, A. and Summers, E. (2009). SKOS Simple Knowledge Organization System Primer. http://www.w3.org/TR/skos-primer/. Retrieved 2015-04-20.

ISO (2014). ISO 10303-242:2014 - Industrial automation systems and integration – Product data representation and exchange – Part 242: Application protocol: Managed model-based 3D engineering.

Kifer, M. and Boley, H. (2013). RIF Overview (Second Edition). http://www.w3.org/TR/rif-overview/. Retrieved 2015-04-20.

Lindemann, U. (2009). Vorgehensmodelle, Grundprinzipien und Methoden. In *Methodische Entwicklung technischer Produkte*, chapter 3. Springer-Verlag Berlin Heidelberg, Garching, Germany.

Matthes, D. (2011). *Enterprise Architecture Frameworks Kompendium*. Springer Berlin Heidelberg.

Meerkamm, H. (2011). Methodology and Computer-Aided Tools - a Powerful Interaction for Product Development. In Birkhofer, H., editor, *The Future of Design Methodology*, chapter 5, pages 55–65. Springer-Verlag London Limited.

Merriam-Webster.com (2015). 'method'. http://www.merriam-webster.com/dictionary/method. Retrieved 2015-04-20.

Müller, J. (1990). *Arbeitsmethoden der Technikwissenschaften: Systematik, Heuristik, Kreativität*. Springer Verlag.

Omrane, N., Nazarenko, A., Rosina, P., Szulman, S., and Westphal, C. (2011). Lexicalized Ontology for a Business Rules Management Platform: An Automotive Use Case. In Olken, F., Palmirani, M., and Soltara, D., editors, *Rule - Based Modeling and Computing on the Semantic Web*, volume 7018, pages 179–192. Springer Berlin Heidelberg.

Ortmann, J., Diefenthaler, P., Lautenbacher, F., Hess, C., and Chen, W. (2014). Unternehmensarchitekturen mit Semantischen Technologien. *HMD Praxis der Wirtschaftsinformatik*, 51:616–626.

Oxford Dictionaries (2015). 'method'. http://www.oxforddictionaries.com/definition/english/method. Retrieved 2015-04-20.

Pahl, G., Beitz, W., Feldhusen, J., and Grote, K. H. (2007). *Pahl/Beitz Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung*. Springer Berlin/Heidelberg, 7th edition.

Rosina, P. and Kiss, E. M. (2011). D4.3 - AUDI R&D Business Orchestration System. ONTORULE Project.

The Open Group (2011). *TOGAF Version 9.1*. Van Haren Publishing, Zaltbommel, Netherlands.

The Open Group (2013). *ArchiMate 2.1 Specification*. The Open Group, Berkshire, United Kingdom.

W3C (2007). Semantic Web layercake diagram. http://www.w3.org/2007/03/layerCake.png. Retrieved 2015-04-20.

W3C OWL Working Group (2012). OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl2-overview/. Retrieved 2015-04-20.

W3PM (2009). Product Modelling using Semantic Web Technologies. http://www.w3.org/2005/Incubator/w3pm/XGR-w3pm/. Retrieved 2015-04-20.

Weigt, M. (2008). *Systemtechnische Methodenentwicklung*. PhD thesis, Universität Karlsruhe (TH).

Wood, D., editor (2010). *Linking Enterprise Data*. Springer-Verlag New York, Inc., New York, NY, USA.