Towards Common Ground in SME: an Ontology of Method Descriptors

Adrian Iacovelli and Carine Souveyet

University Panthéon Sorbonne. 90 rue Tolbiac,
75013 Paris, France.
{adrian.iacovelli, souveyet}@univ-paris1.fr

Abstract. The Method Engineering (ME) community is a prolific research domain where competing Situational Method Engineering (SME) approaches have been defined and used for composing, adapting or/and configuring a method into modular constructs according to their own modularization vision. This diversity shows the richness of the ME domain but implies some drawback like unnecessary confusion for non ME expert, lack of standard & interoperability, lack of implementation tool. However, researchers are agreed that a common ground in SME is a hot matter of discussion. Assuming that the differences between SME approaches are purposeful, we propose to reach a semantic common ground on what types of core concepts constitute a method descriptor. To achieve it, an ontology-based approach is applied in SME to design an ontology of method descriptors as a domain ontology. The semantics of the six most popular SME approaches modular constructs are defined according to this ontology in order to show its usage and its relevance. Finally, usage scenarios have been sketched to show that the ontology can be the start up phase for reducing the ME drawbacks mentioned above.

Keywords: Method Engineering, Method Descriptors, Ontology, Service Oriented Architecture.

1 Introduction

Information systems development methods are the subject of study of Method Engineering (ME) science. One of the ME interests is to decompose into modular parts these methods for optimizing, reusing, and ensuring their flexibility and their adaptability [1]. This interest is the basis of the Situational Method Engineering (SME) community. This domain is a prolific research domain where several competing SME approaches have been defined, published and used with their own vision of method modularization. This diversity shows the richness of the research works but implies some drawbacks like unnecessary confusion for non ME experts [1], lack of standard & interoperability, lack of implementation tool [2].

Today, a common ground in ME is a hot topic of discussion between researchers [1]. According to the authors of [1], there are two possible solutions: (1) differences are minor and an agreement on what modular construct to promote can be reached, or

(2) the diversity is useful because they serve different purposes and there is a need for them to co-exist.

In the past, we had published a framework for the method modular constructs comparison for underlying their semantic differences [2, 3], and pushed our vision for a specific matter. However, today, we believe and assume that the diversity is purposeful. But we also believe that a semantic common ground in SME is needed and can be achieved. This semantic common ground can be considered as a start-up phase to reduce directly or indirectly drawbacks such as (a) unnecessary confusion for non ME expert, (b) lack of standard & interoperability, and (c) lack of implementation tool.

The purpose of this paper is to propose an ontology-based approach to design the semantic common ground in SME and sketches its benefits into exploring scenarios to reduce the drawbacks mentioned above. An ontology was proposed in the ME field in [4] to define the core concepts required for qualifying knowledge about method. But it is a top lightweight ontology which not allows to define a common ground for SME approaches. Another ME based ontology was proposed in [5] but their concepts are too restrictive to cover the diversity of method modular constructs and levels of granularity introduced in the various SME approaches. In addition, their objective is to improve the SME approach proposed in [6] and not to find a common ground in SME.

In a philosophical point of view, ontology is the study of the categories of things that exist or may exist in a particular domain. In other words, domain ontology defines the types of things in that domain. Moreover, ontology is a fundamental part of the knowledge, and all other knowledge should rely on it or refer to it.

SME approaches [7, 8, 9, 11, 11] promote different modular constructs of a method but they have a common understanding of what a method is [12, 13]. Here, a method is described by five interrelated ways: a way of thinking (paradigm), a way of modelling (product), a way of working (process), a way of supporting (tool) and a way of controlling (organisation). Therefore, the core of ME is represented by the common understanding of the various things that constitute or may constitute a method description. Consequently, method descriptors ontology is designed as domain ontology and the various modular constructs of SME approaches such as ‘method fragment’, ‘method chunk’, ‘method component’, ‘process component’ are defined semantically by referring the concepts of the ontology.

The paper is organised as follows. The ontology of method descriptors is explained in section 2. Section 3 illustrates how various SME modular constructs match the ontology. Furthermore, exploring usage scenarios of the Ontology are sketched in section 4 to illustrate the interest of the ontology and to explore future research options for reducing the ME drawbacks mentioned above. Finally, section 5 concludes this work with our contribution and research perspectives.

2 Method Descriptors Ontology

This section explains the concepts and their relationships defined in the Ontology of method descriptors. The SME approaches have different method modular constructs but they agree on the understanding of what a method is [12, 13]. According to this definition, a method is designed as a collection of method modular constructs. Figure 1 illustrates our ontology is built upon this definition.

The reminder of this section explains the concepts defined in the Ontology of Method descriptors and their relationships.

The following top concepts of the ontology illustrated in Figure 1 are rooted to the "*Thing*" concept : *Method Puzzle*, *Goal*, *Verb*, *Target*, *Parameters*, *Paradigm*, *Concept*, *Modelling Element*, *Modelling Rule*, *Annotation*. Moreover, "is a" links of the ontology define specialisation relationships between two concepts.

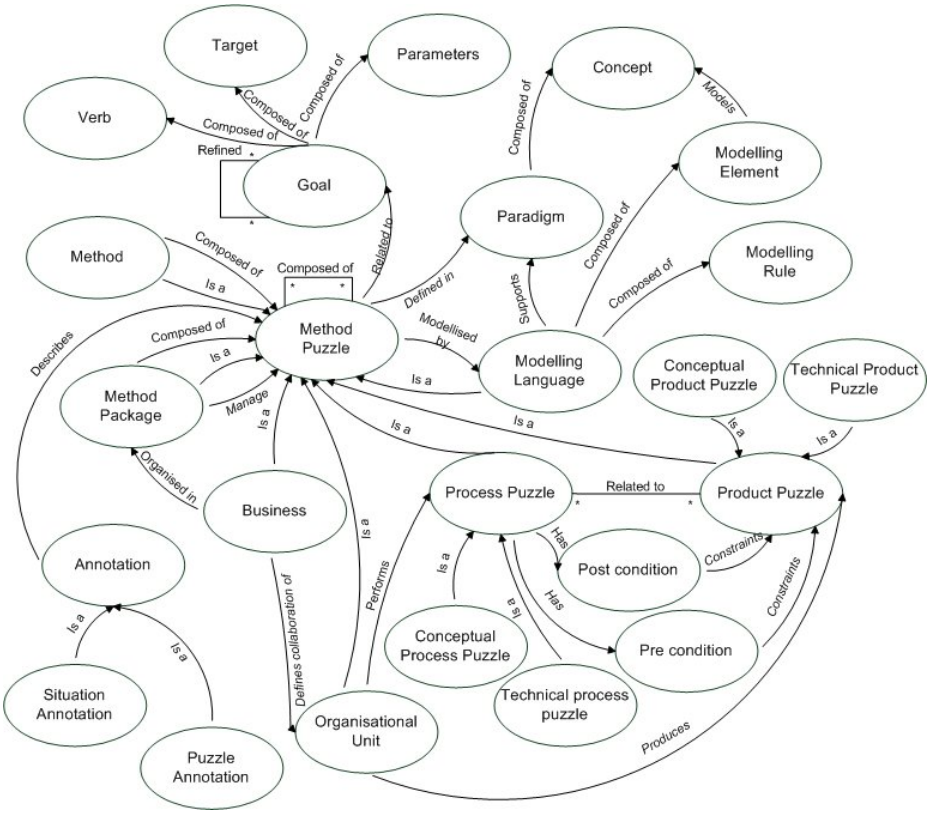

Fig. 1. Method Descriptors Ontology

The word *method* comes from the Greek *methodos* which means way of investigation [7]. According to Harmsen a method is an integrated collection of procedures, techniques, product descriptions and tools for an effective and efficient support on the

engineering process [14]. In the situational method engineering context, the method modular constructs (i.e. *Method Puzzle*) are assembled to produce a method tuned to its application situation. This decomposition into modular *Method Puzzle* parts is fundamental for the flexibility, adaptability, optimization and reuse of methods [15]. As a composition of *Method Puzzle*, a *Method* is also viewed as a *Method Puzzle*.

A *Method Package* is an autonomous reusable part of a *Method* capturing one of its particular aspects. It defines an assembly of *Method Puzzles* responding to a specific kind of project. It is either a preconfigured part of a *Method.* Alternatively, it can incorporate a temporal dimension to organize its composing *Method Puzzles* [8, 16, 17].

A *Method Puzzle* describes an element of a method. It's a coherent piece of an IS development method [7]. In order to manage the complexity of a method definition, SME approaches emphasize a modular vision of its definition. This vision is introduced by the *Method Puzzle* concept. *Method Puzzles* can be defined at different level of granularity, i.e. *Method Puzzles* can be composed of other *Method Puzzles*. In addition a *Method Package* is viewed as a composition of *Method Puzzles* and is also considered as a *Method Puzzle*. Notice that the two specializations of a method puzzle: method and method package are exclusive with all other specializations. A *Method Puzzle* specialized into a *Method* or a *Method Package* can't have any other specialization. The other *Method Puzzle* specializations correspond to the Seligmann and Rolland [12, 13] method definition: *way of thinking, a way of modeling, a way of working, a way of control and a way of supporting*. The way of thinking is the philosophy used in the *Method Puzzle* which is captured in the *Paradigm* concept and supported by a *Modeling Language*. The way of modeling describes the various constructs and their models related to the method application. This way is defined into the *Product Puzzle* concept and more precisely by the *Conceptual Product Puzzle* concept. To complete the way modeling definition, the way of working expresses how to perform a method or how a product evolves during a method. This way is identified in our ontology by the *Process Puzzle* concept and more particularly by the *Conceptual Process Puzzle* concept. The way of control specifies how to organize the performing of a method process into an organization and is described by the *Business* and *Organizational Unit* concepts. The way of supporting is the tool for supporting the method. It is related to the *Technical Process Puzzle* and to the *Technical Product Puzzle* concepts. The various SME approach share the same view of a *Method* but have different definitions of a *Method Puzzle*. To take into account this diversity, the various specialization of *Method Puzzle* such as *Product Puzzle*, *Process Puzzle, Modeling Language, Business* and *Organizational Unit* are inclusive. For example a *Method Puzzle* can be specialized into both *Process* and *Product Puzzles* at the same time like in the Chunk Approach [11].

The main purpose of a method puzzle is to be reused in different methods. In order to increase its reusability, one has to provide a mechanism for extracting and regrouping the key concepts of a method puzzle. This concise information on a method puzzle is called annotation. It helps searching and retrieving method puzzle. Two types of information are required: (a) information regarding the situations where a *Method Puzzle* can be reused and (b) information to characterize and summarize the

content of a *Method Puzzle*. The first is managed by the *Situation Annotation* and the second is handled by the *Puzzle Annotation*.

A *Method Puzzle* helps to achieve a particular *Goal*. A Goal in this case is a statement expressing what is wanted [18]. I.e. it represents the state to be reached or maintained. A linguistic approach proposed in [19] and its extension in [20] are based on the case grammar. They recognize a goal statement as a combination of a verb, a target and parameters. A goal verb is the central component of the statement. It describes the action to be performed. The target is the subject of a goal statement. It can depict the expected result of a goal achievement or an existing entity modified by performing a goal. In addition parameters are complementary information exposed in a goal statement. In fact, each parameter plays a semantic role according to the verb. Goals can be defined at different levels of granularity. It means that the achievement of a complex goal may require the achievement of sub-goals. We say here that a goal can be refined by a set of (sub) goals. In the ontology it is expressed by the recursive *refined* relationship of the goal concept.

Paradigm is a coherent model of a world perception grounded on a specific philosophy. A *Paradigm* describes a set of concepts and their interactions that cannot be mixed with another *Paradigm*. In our ontology, concepts used to express the paradigm are introduced by the *Concept* node**.** This node helps to represent the constructs and their relationships useful for the paradigm description. A Modeling Language is used to design world according to the concepts of the paradigm. It is considered as a tool to produce models. A Modeling Language can be itself defined as a *Method Puzzle*: a set of *Modeling Elements* defined according to *Modeling Rules*. A *Modeling Element* is a textual or graphical representation of a *Concept* from a *Paradigm* whereas a *Modeling Rule* is an axiom that must be satisfied by modeling elements or a set of *Modeling Elements.*

In SME approaches, the *Process Puzzle* is one of the key concepts. It is an abstract element aimed to capture a process for achieving the *Method Puzzle Goal* [10]. It is the work that has to be done in order to obtain the result [17]. Or, more precisely, it is the set of actions which transforms a product (*Product Puzzle*) under development [11], from a source product to a target product [8]. As a *Process Puzzle* is a specialization of a Method Puzzle, it can be defined at various level of granularity. So it can describe high-level project strategies or more detailed development procedures [7]. The modeling of this process structure is supported by the *Conceptual Process Puzzle* concept. This concept includes a set of process descriptions and models. Its implementation is supported by the *Technical Process Puzzle* concept. This concept represents an operational tool automation of the *Process Puzzle*. As shown in [10], two other concepts are related to the *Process Puzzle*: a *Precondition* and a *Postcondition*. The Precondition concept defines an initial situation required for applying a *Process Puzzle*. It is a restriction constraining the input *Product Puzzles* instances of a *Process Puzzle*. A *Precondition* defines the expected state of *Process Puzzle* input products. At the opposite, a *Postcondition* concept defines a final situation resulting of the application of a *Process Puzzle*. It is a restriction constraining the output *Product Puzzles* instances after the performing of a *Process*

Puzzle. The *Postcondition* defines the expected state of Process Puzzle output products.

Another key concept of SME approaches is the *Product Puzzle*. It's an abstract element capturing the product aspect of methodologies [17] and it conforms to the paradigm adopted in the methodologies. A Product Puzzle models artifacts used or produced by the performing of a Process Puzzle [7, 17, 10, 11]. These artifact models are defined as *Conceptual Product Puzzle* concept whereas their instance implementations are supported by *Technical Product Puzzle* concept.

The way of control identified above is represented by the *Business* and *Organizational unit* concepts. An *Organizational Unit* is a resource, an actor role or a set of actors (team or bigger groups) description involved into performing of a *Process Puzzle* in order to produce a product described in a *Product Puzzle*. A *Business* is used to model the collaboration of *Organizational Units*. It captures the interactions between *Organizational Units* in order to perform a project or a business mission of an enterprise [21]. A Business is related to several *Method Packages* and temporally organized into them. The business concept can be also considered as a Method puzzle.

In this section the core concepts of a SME descriptor that constitute the ontology have been explained. The next section illustrates how these concepts are used to define the semantics of modular constructs of selected SME approaches.

3 SME Method Descriptors

This section illustrates how our ontology defines semantics for each SME modular construct. We have selected the five most cited component-based SME approaches such as [7, 8, 9, 10, 11] and one approach defined in the service orientation. We show that the ontology can constitute a common ground in SME approaches which are component or service based approaches.

3.1 Method Fragments

In [7], Brinkkemper and colleagues propose the method fragment concept. This concept is one of the earliest modularization constructs in ME [1]. According to [7] method fragment is a standardized building block based on a coherent part of a method [7]. It is an abstract element defined at one of the five different layers of granularity: method, stage, model, diagram, or concept [22]. The method fragment concept matches with the *method puzzle* concept as it is defined in our ontology and its granularity is captured by the composition link. A fragment can be specialized either into a process fragment or a product fragment (cf. Figure 2). As product fragments models the structures of the methods products and process fragments are models of the development process [7], they can be respectively defined as a specialisation of the *conceptual product and process* puzzles concepts.

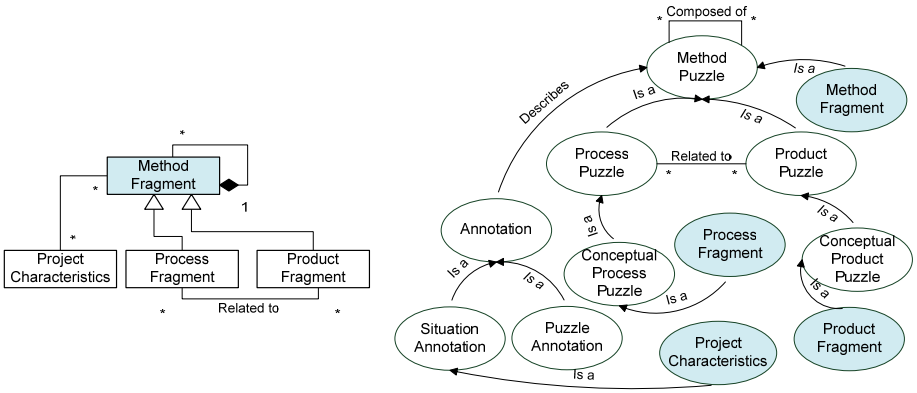

Fig. 2. Method Fragment Structure and Semantic Matching

This specialisation is a specific case of the ontology method puzzle as it's an exclusive specialisation into product or process fragments. The retrieval and use of method fragments is provided by project characteristics attached to each fragments. These situation characteristics match with the situation annotation concept of the ontology.

3.2 Method Chunks

The method chunk approach was proposed by Rolland and colleagues [11]. A method chunk is organised into two levels of knowledge: a method knowledge level and a meta-knowledge level [1, 23]. The method level of the method chunk concept is driven by its method process part which is attached the product part. As a method chunk is a composition of one process part and one product part, the method chunk is characterized by an inclusive specialisation into both a *conceptual process puzzle* and the *conceptual product puzzle*. The process and product part are defined respectively in our ontology as a specialisation of the *conceptual process puzzle* and the *conceptual product puzzle* with a more specific one to one cardinality on the relationship between their *process puzzle* and *product puzzle* parent concepts. The body concept in the method chunk approach is an abstract concept design for the encapsulation the process and product part. As it's an abstract concept that doesn't support additional new semantic to the chunk concept, there is no need to model it in the ontology. The interface of a method chunk captures information on the chunk and its goal. The matching of the interface concept in our ontology is done by a double inheritance link with the *puzzle annotation* and the *goal* concepts. In the same way, the meta-knowledge level of method chunks captured by the descriptor concept is defined with a double inheritance link the *situation annotation* and the *goal* concepts.

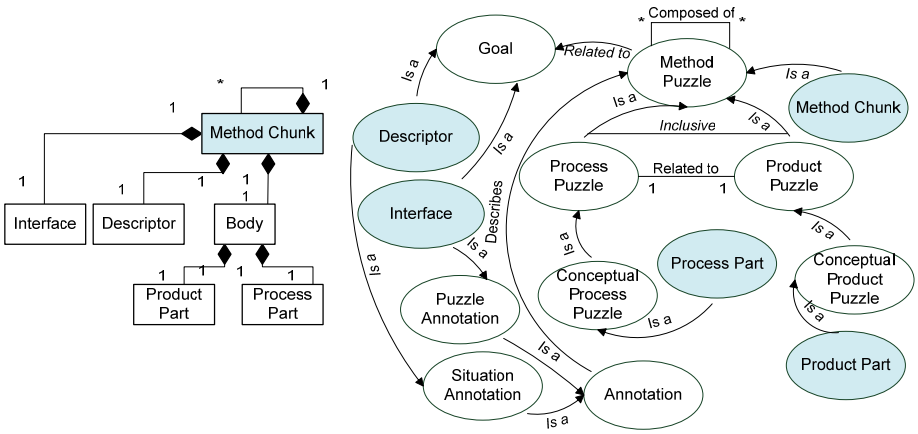

Fig. 3. Method Chunk Structure

The purpose of the descriptor concept is to capture the situational aspects of method chunks usage to support their retrieval process. Descriptor contains the goal definition of the chunk and a set of parameters characterising the situation of reuse of this chunk.

3.3 Method Components

The method component concept aims to capture a self-contained part of a system engineering method [8]. The latest contributions to this concept were made by Karlsson and colleagues in [8, 16]. Method component is designed to be used in a specific kind of ME, the method configuration. Each component has to address a certain aspect of the problem at hand and it is the smallest part of a method that is practically useful [1]. For these reasons the method component concept is mapped to the method package concept of our ontology. A method component is build by an assembly of several method elements that are the basic constructs constituent of a method: action, artefact, actor role, concept and notation. This method element concept can be defined as a specialisation of the method puzzle concept. An action is the set of tasks to be performed during the method component application. As actions are the central constituents of the method process model [8] they can be defined as a specialisation of the *conceptual process puzzle concept* in our ontology. The results of these actions are represented by artefacts in the method product model [8]. The artefact concept is matched with the conceptual *product puzzle concept*. Furthermore, the actions are performed by project members who have different roles during the project [8], this implies that the actor role concept can be mapped with the organisational unit concept of the ontology. A set of concepts is used to describe the problem domain of the method component and they are captured and represented using notations [8]. These concepts respectively correspond to the *concept* concept and the *modelling element* concept of the ontology. Both method components and method elements are linked to goals which can be refined in sub-goals. That defines a

perfect match between the goal concept form the method component approach with the goal concept of the ontology.

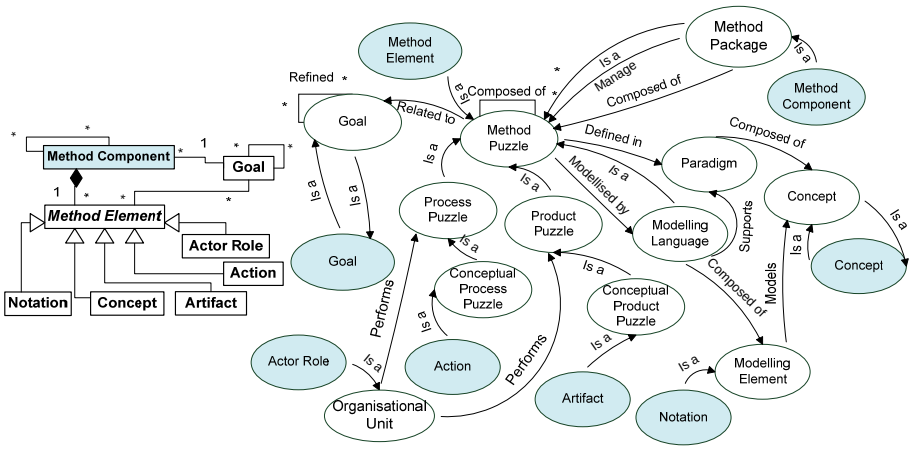

Fig. 4. Method Component Structure

3.4 Open Process Framework (OPF) Method Elements

Based on the international standard ISO/IEC 24744 [24] the OPF approach was proposed by Henderson-Sellers and colleagues [9], the last updates of the approach can be found in [21]. Each OPF method component is generated from an element in a prescribed underpinning meta-model [1] according to the ISO standard. An OPF method component is defined as an abstract element which all other method constructs are derived [21]. So it can be defined as a specialisation of the *method puzzle* concept in our ontology. The OPF approach is driven by the decomposition of the method process in work units performed by producers known as people role and teams. These two latter concepts match respectively with the *conceptual process puzzle* and the *organisational unit* concepts of our ontology. Various products are used or created by work units in order to deliver the final system [17]. This product aspect of methodologies is captured in the work product concept of the OPF approach that can be defined as a specialisation of the *conceptual product puzzle* of the ontology. The work products are documented using a language consisting in a "vocabulary" and a set of "grammatical rules" [21]. These latter concepts can be mapped respectively with our *modelling language*, *modelling element* and *modelling rule* concepts. All these OPF method components are used during a stage and performed by a specific collaboration organisation of producers called a endeavour [21]. A stage models the intended timing of the performance of a temporally-cohesive set of work units during the enactment of a method [21] and can be defined as a specialisation of the ontology *method package* concept whereas the endeavour concept can be defined as a specialisation of the *business* concept.

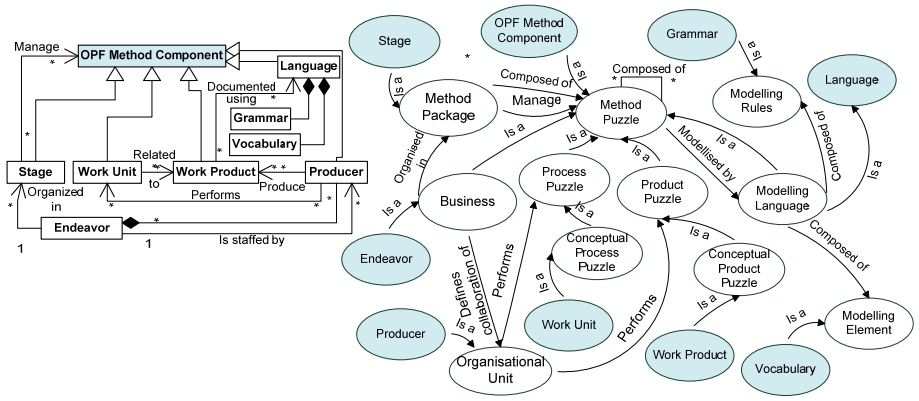

Fig. 5. OPF Method Component Structure

3.5 SO2M Method Services

Introduced by Guzelian and colleagues [10] the SO2M approach is the first step of applying the service oriented paradigm [25] to ME approaches.

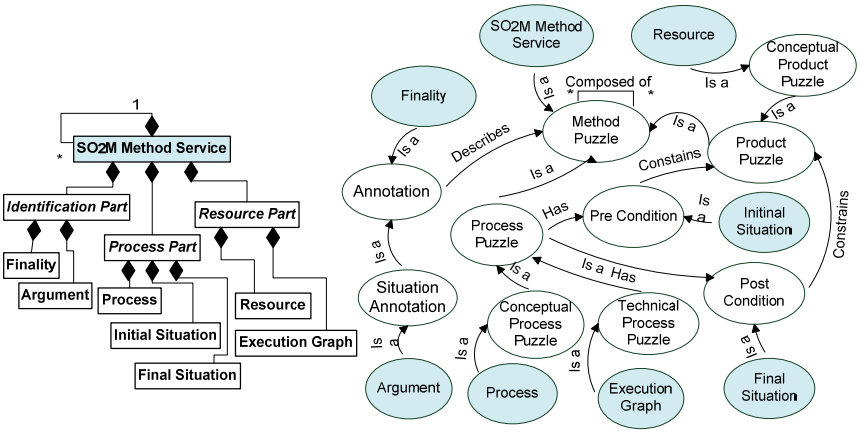

Fig. 6. SO2M Method Service Structure

A SO2M method service is a reusable unit that contains one or several method fragment to solve an information system development problem [10]. It can be mapped with the *method puzzle* of the ontology and exploit the inclusive property of the *method puzzle* specializations possibilities. A method service is constituted of three abstract parts: identification part, process part and resource part. As these parts are just abstract containers they won't be match with the ontology.

The identification part aims to capture the contextual knowledge of the method service reuse by defining its finality and argument. A finality is the description of the problem solved by a method service, it's structured with a goal, a manner and a

context [10]. As it contains both situational and structural information the finality concept can be defined as a specialization of *annotation concept* in our ontology. The argument concept of the SO2M approach characterizes a method service reuse situation by a list of pro arguments (i.e. advantages) and con arguments (i.e. drawbacks) and it can be matched with the ontology *situation annotation* concept.

The process part is composed of the process initial situation, process final situation and process structure description. These three concepts are respectively matched in our ontology with the *precondition*, *postcondition* and *conceptual process puzzle* concepts.

The resource part defines the implementation of a process by an execution graph which can be mapped as a specialization of our *technical process puzzle* concept. This part also defines the descriptions of all resources consumed or delivered by the process. This latter concept can be defined as a specialization of the *conceptual product puzzle* in our ontology.

This section shows that each concept of the method descriptor ontology is matched in the set of concepts issued of the five studied SME approaches and acknowledges the relevance of the ontology. Furthermore, the matching between the studied approaches and the ontology of method descriptors shows that each of these approaches shares common concepts with the others and also incorporates new concepts to characterize method constructs not addressed in the others. The ontology represents a semantic common ground useful to understand the semantic difference between the various SME approaches. To emphasize the benefit of this ontology and in particular in reducing the ME drawbacks mentioned earlier, three exploring usage scenarios are sketched in the next section.

4 Exploring usages of the Method Descriptors Ontology

The ontology of method descriptors is an attempt to reach a semantic common ground in SME. This section explores possible usage scenarios of such ontology. Four usage scenarios have been envisioned and described to illustrate the relevance of the ontology-based approach and its usefulness for future ME perspectives.

1. The ontology can be used in an educational manner by non ME experts to understand (i) the basic semantic common ground of the domain and (ii) the various competing modular constructs proposed in SME. This basic usage helps directly in reducing the first ME drawback (confusion).
2. The ontology can be used as a basis of ME reasoning systems such as decisional support system helping ME engineers in their tasks. This usage is complementary to the educational usage.
3. In addition, the ontology can a first step of a process building a unified ME query facility on top of unified Method knowledge Base. Such ME tool is helpful to ME engineers to extract ME knowledge according to their needs expressed in a common language (ontology concepts). Then, a mapping facility must be built to translate the initial query into a specific query compliant to the ME descriptor of the method base. The ontology allows building a generic tool to query method base

storing method puzzles belonging to several SME approaches or to query various method bases compliant to SME approaches. Finally, it is a way of reducing ME drawback like lack of SME standard. In fact, the main interest is not in the language used to describe the method puzzle but the fact that the method puzzle matches the ME engineers needs. The perspective of this usage is to propose to the ME community to build a common Method knowledge base which can become a reference for the community and the practitioners.

4. Service orientation in Information Systems leads to re-organize a portfolio of legacy applications into services. By analogy to Information System engineering, we can assume that a service orientation in the ME, leads to re-organize CASE tool into method services (end-user software service). To adapt services oriented technologies to method services, we should extend the service descriptor as it is sketches in Fig. 7.

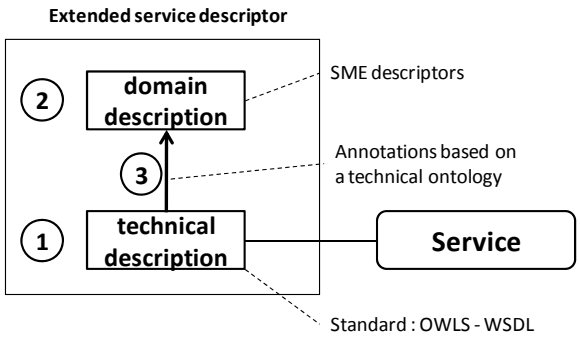

Fig. 7. Principe of service description extension

Figure 7 summarizes YASSA approach [26] to extend the service description to domain description with the usage of a technical ontology. The service description is structured in two layers: technical service description based on standards such as OWLS or WSDL and a domain description represented in our case by any SME descriptor encoded in an XML format. The technical service description refers the domain descriptor through annotations embedded in the technical layer and referring domain value mentioned in the domain description. Annotations are expressed according to the ontology concepts in order to build search algorithms according to the ME ontology instead of the SME descriptor. It means that a Method Service registry is built upon the semantic common ground in SME instead a specific SME approach. This usage allows encapsulation of a tool support part to any SME descriptor with searching algorithms ME ontology compliant but SME descriptor independent. This usage scenario shows a way of reducing indirectly the third ME drawback (lack of implementation tool).

Assuming that a CASE tool is re-organized as a portfolio of method services, and a CASE tool is considered as configurable tools like ERP and product line, the perspective is to configure the method part and the case tool supporting it, at the same time. By analogy to the Product line, this perspective introduces the concept of a

Method Line where method is compliant to Seligman definition and the purpose of the CAME is to configure the method and the CASE tool support at the same time. In this case, the CASE tool can be viewed as an assembly of method services which can be combined into a specific configuration.

This section is illustrating how the method descriptors ontology may be used and how it is possible to reduce directly or indirectly the ME drawbacks. These usage scenarios show that a semantic common ground may be enough to step towards in the ME community.

5 Conclusion

In this paper, we have assumed that the diversity in SME approaches is purposeful and shows the richness of the ME community. It is largely agreed that a common ground is needed to overcome some ME drawbacks such as unnecessary confusion for non ME expert, lack of standard & interoperability and lack of implementation tool, but is also a hot topic between researchers. In addition, SME approaches have not been yet largely used by practitioners, or implemented in CAME environment because of these ME drawbacks. The paper proposed an ontology-based approach in SME to build the ontology of method descriptors as a domain ontology. SME approaches promote different method modular constructs but they have a common understanding of what a method is. We exploited the Seligman definition of a method : way of thinking (paradigm), way of working (process), way of modelling (product), way of controlling (organisation) and way of supporting (tool support). Therefore, the ontology defines the core concepts of a method description and the granularity levels built upon them. We assumed that this ontology constitutes a semantic common ground in SME which is a start-up phase in reducing indirectly the ME drawbacks.
However, to be effective, the SME approaches must define their semantic according to the ontology. We showed in this paper how the ontology can be used to define the semantic of the six most cited SME approaches : ‘Method Fragments’, ‘Method chunks’, ‘Method components’, ‘OPF method elements’ and ‘SO2M method services’. Then, the ontology of method descriptors obtained showed that SME approaches shared common concepts but also incorporated new concepts to characterize methods constructs not addressed in the others. It is why we assumed that differences between SME approaches are purposeful and we have adopted an alternative solution: semantic common ground.

Finally, the paper explored three usage scenarios of the ontology of method descriptors. The ontology can be used as an educational tool for non ME expert to reduce their confusion or as a basis of reasoning systems.
A step forward, the ontology can be used to build a unified ME query facility. In fact, the ontology is used as a mapping tool between the ME engineers query and the technical query executed on a specific method base compliant to a specific SME approach. The benefit of this usage can be to develop one multi-approach method knowledge base which can be the reference of the ME community and can be shared with practitioners.

Moving SME approaches to service orientation, implies to move the CASE tool in the centre of a method description. We illustrated in the paper the application of YASSA's approach to extend service to method service and service description to method service description. A method service description is composed of two related layers: technical and domain service description (ME descriptor) layers. The ontology of method descriptors is used to integrate ME annotations inside the technical service description conform to standards like OWLS or WSDL and it allows to provide searching algorithms of method service built upon the Ontology of Method descriptors instead of the SME descriptor itself. The method service can be described at the domain layer by any SME descriptor.

Finally, the service orientation combines with the ERP or Product line analogy, we can envision the CASE tool and its method description as a method line and its objective is to provide CAME to configure the method description part and the CASE tool at the same time. The perspective is a subject of research.

References

1. Agerfalk, P., Brinkkemper, S., Gonzales-Perez, C., Henderson-Sellers, B., Karlsson, F., Kelly, S., Ralyté, J.: Modularization Constructs in Method Engineering: Towards Common Ground?, Panel of ME'07, Springer, Geneva, Switzerland, (2007)
2. Deneckère, R., Iacovelli, A., Kornyshova, E., Souveyet, C.: From Method Fragments to Method Services. In: EMMSAD Workshop of CAISE'08, Montpellier, France (2008)
3. Nehan, Y.-R., Deneckère, R.: Component-based Situational Methods: A framework for understanding SME, in IFIP, Vol. 244, Situational Method Engineering: Fundamentals and Experiences, Switzerland, (2007)
4. Mirbel, I.: Connecting Method Engineering Knowedge: a Community Based Approach. In: proceedings of ME'07, Geneva, Switzerland, (2007)
5. Niknafs, A., Asadi, M., Abolhassani, H.: Ontology-Based Method Engineering. In: International Journal of Computer Science and Network Security. IJCSNS. vol. 7, 8 (2007)
6. Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: proceedings of the international conference CAISE'03, Austria (2003)
7. Brinkkemper, S.: Method Engineering: engineering of information systems development method and tools. Information and Software Technology, vol. 38, 7 (1996)
8. Wistrand, K., Karlsson, F.: Method components: Rationale revealed. In: proceedings of CAISE'04, Springer-Verlag. Riga, Latvia, (2004)
9. Henderson-Sellers, B.: Process meta-modelling and process construction: examples using the OPF. Ann. Software Engineering, vol. 14, 1-4, (2002)
10. Guzélian, G., Cauvet, C.: SO2M : Towards a Service-Oriented Approach for Method Engineering. In: proceedings of the international conference IKE'07, USA, (2007)
11. Rolland, C., Plihon, V., Ralyté, J.: Specifying the reuse context of scenario method chunks. In: proceedings of the international conference. CAiSE'98, Pise, (1998)
12. Seligmann, P.S., Wijers, G .M., Sol, H.G.: Analysing the structure of IS methodologies, an alternative approach. In: proceedings of the 1st Dutch conference on Information Systems, Amersfoort, The Netherlands, (1989)
13. Plihon, V. ,Rolland., C.: Modelling Ways–of–Working. In: proceedings of CAISE'95, Springer-Verlag, Jväskylä, Finland (1995)
14. Harmsen, AF.: SituationalMethod Engineering. Moret Ernst & Young.(1997)

15. Rolland, C.: in French: L'ingénierie des méthodes : une visite guidée. In: e-TI. (2005)
16. Karlsson, F., Agerfalk, P.J.: Method configuration: adapting to situational characteristics while creating reusable assets. In: Information Software and Technology, vol. 46 (2004)
17. Gonzalez-Perez, C.: Supporting Situational Method Engineering with ISO/IEC 24744 and the Work Product Pool Approach. In: IFIP, Situational Method Engineering: Fundamentals and Experiences (2007)
18. Jackson, M.: Software Requirements & Specifications – a Lexicon of Practice, Principles and Prejudices. ACM Press, Addison-Wesley (1995)
19. Fillmore, C.J.: The case of case. In: Universals in linguistic theory, Holt, Rinehart and Winston Inc. (1968)
20. Dik, S.C.: The theory of functional grammar, Foris Publications, The Netherlands (1989)
21. Open Process Framework, http://www.opfro.org/
22. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-Modelling Based Assembly Techniques for Situational Method Engineering. Information Systems, vol. 24, 3, (1999)
23. Rolland, C., Prakash, N.: Aproposal for context-specific method engineering. In: Principles of method construction and tool support, Chapman & Hall, vol. 191-208 (1996)
24. International Organization for Standardization: ISO/IEC 24744, Software Engineering – Metamodel for Development Methodologies.
25. Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering WISE'2003 (2003)
26. Chabeb, Y., Tata, S.: Yet Another Semantic Annotation For WSDL. In: Proceeding of International Conference WWW/Internet IADIS'08 (2008)