AOMIP -105
An approach based upon OWL-S for method fragments documentation and selection

ABSTRACT

It is currently admitted in both mainstream software engineering and agent-oriented software engineering that there is no one-size-fit-all methodology or process. One solution is proposed by the situational method engineering paradigm that provides means for constructing ad-hoc software engineering processes following an approach based on the reuse of portions of existing design processes, the so called method fragments, stored in a repository called method base. One problem raised by this type of approaches is to describe or document fragments and to choose among existing fragments in order to build a new process. The approach proposed in this paper uses OWL-S and documents fragments as services in order to semantically annotate methodologies concepts and fragments. This approach enables method engineers to select fragments from an existing method base by using semantic reasonings based on an ontological description of fragments and queries. A scenario illustrating an example of fragment selection is reported. The case study uses a simple ontology and the support of the Protg tool.

1. INTRODUCTION

It is currently admitted in mainstream software engineering and agent software engineering that there is no one-size-fit-all methodology or process. Indeed, as stated in [19] "traditional rigid IS engineering methods are inadequate to provide the necessary support in new IS developments. New methods, more flexible and better adaptable to the situation of every IS development project, must be constructed".

One solution is proposed by the situational method engineering paradigm. Situational method engineering paradigm provides means for constructing ad-hoc software engineering processes following an approach based on the reuse of portions of existing design processes, the so-called method fragments, stored in a repository, called method base.

One problem raised by this type of approaches is to describe or document fragments and choose among existing fragments in order to build a new process. This problem is reinforced by the fact that different methodologies may use different concepts, different processes and different guidelines. This is obviously one of the hypothesis of SME. Fragments are thus heterogeneous. Designing a notation to describe these heterogeneous fragments that ease the search of a best fitting fragment for a given purpose is a tough matter.

In order to ease the work of the method engineer we propose an approach based on the adoption of OWL-S [16] as a description of fragments. The idea is to compare each fragment with a service and use the three ontologies encompassed by OWL-S to describe a fragment. Indeed, the Service Profile ontology can describe the inputs and outputs elements of the meta-model underlying the fragment and the result of the fragment can be described by the service result. The Service Model ontology of OWL-S can describe the fragment tasks and eventually the Service Grounding ontology can state what are the fragment guidelines and use possibilities. The advantages of such an approach are twofold. Firstly, it enables to describe, using ontologies, the semantics between each fragment and their related methodologies. Secondly it allows a method engineer to use the matchmaking mechanisms allowed by OWL-S. These two advantages may answer to how to describe a fragment and how to choose the best fitting fragment questions.

In this paper we illustrate the use of OWL-S for fragment documentation and selection through a scenario in which a method engineer search to extend an existing methodology, namely ASPECS [8], with fragments allowing requirements decomposition. These fragments, in the scenario, will come from the TROPOS methodology [2].

This paper is organised as follows: section 2 introduces the basics for understanding OWL-S, section 3 illustrates our approach with a fragment selection scenario, section 4 presents some related works and section 5 concludes.

2. BACKGROUND

2.1 OWL-S

The OWL-S ontology [16] is composed by three sub-ontologies: the service profile, process model, and grounding. The profile sub-ontology or profile is used to describe what the service does. The process model sub-ontology or process model is used to describe how the service is used. The grounding sub-ontology or grounding is used to describe how agents can interact with the service. Each service described using OWL-S is represented by an instance of the OWL class Service.

The OWL-S profile enables to advertise what a service does and/or dually what is expected from a service. A profile is an instance of the class Profile. A Profile describes the

```
<service:Service  rdf:ID="CurrencyConverterService">
  <service:presents rdf:resource="#CurrencyConverterProfile"/>
   <service:describedBy  rdf:resource="#CurrencyConverterProcess"/>
   <service:supports rdf:resource="#CurrencyConverterGrounding"/>
</service:Service>
```

Figure 1: Example of OWL-S class: the CurrencyConverterService from

inputs, outputs, preconditions and results associated to a service. The inputs (resp outputs) are the object descriptions the service takes as input (resp produces as output). The precondition of a service states what must be true before the service is called. The results states what is expected to be true after the completion of the service.
The two classes reported in Figure 1 and Figure 2, extracted from [17], consist in a service and its profile. The service converts a price given with one currency into another currency. The service class gives an id to the service: CurrencyConverterService and relates the service to the three classes describing it: profile, process and grounding.

The profile class defines an id: CurrencyConverterProfile, relates this profile to the service encompassing it and defines the inputs (the given price and currency for convertion), outputs (the price in the wanted currency) and a textual description for human readers.

2.2 ASPECS

ASPECS is a step-by-step requirements to code software engineering process based on a metamodel which defines the main concepts for the proposed HMAS analysis, design and development. It integrates design models and philosophies from both object- and agent-oriented software engineering (OOSE and AOSE) and is largely inspired by the PASSI [4] and RIO [13] approaches. The target scope for the proposed approach can be found in complex systems and especially hierarchical complex systems. The main vocation of ASPECS is towards the development of societies of holonic (as well as not-holonic) multiagent systems.

The idea underpinning the ASPECS design process can be described in a few fundamental choices that characterise the vision of the authors:

1. The ASPECS design process explicitly deals with the design of open, dynamic and complex systems. The main limit in its application scope is that we assume the system can be hierarchically decomposed in subsystems (nearly-decomposable systems [23]).

2. The adoption of an organisational approach. Functionalities to be realised are assigned to organisations that accomplish them also by means of the hierarchical decomposition of the organisation structure in sub-organisations (holonic paradigm). An organisation-oriented analysis is applied by using two different decomposition strategies: vertical and horizontal. Vertical decompositions allows to delegate the responsibility of an organisation at level n to sub-organisations at level n-1 (lower abstraction level, finer grained organisations). Horizontal decomposition allows the collaboration of several entities at the same level of abstraction in order to fulfil the required functionalities.

3. Domain related ontological knowledge is used as a tool for enhancing the quality of design. This has been already adopted in some previous methodologies [15] but it is lacking in most modern approaches. We think that in dealing with intelligent agents it is particularly important to explicitly catch an ontological model of the problem and solution domains; this allows an easy application of several AI techniques as well the adoption of semantic-based communications among agents.

4. Three main levels of abstractions, called models according to the model-driven engineering terminology, are considered. Concepts of the problem domain are used to model system requirements in terms of organisations and interacting roles; concepts of the agency domain are the result of a set of transformations from the previous domain and are used to depict an agent-oriented solution; concepts of the solution domain are again the result of some transformations and are devoted to design a platform-specific solution at the code level.

ASPECS software processes is driven by requirements. Thus the starting activity deals with the analysis of system functional and non functional requirements. Functional requirements describe the functions the software has to exhibit [14] or the behaviour of the system in terms of interactions perceived by the user. Non functional requirements are sometimes known as constraints or quality requirements [14]. The global objective of the Domain Requirements Description (DRD) activity is gathering needs and expectations of application stakeholders and providing a complete description of the behaviour of the application to be developed. In the proposed approach, these requirements should be described by using the specific language of the application domain and a user perspective. This is usually done by adopting use case diagrams for the description of functional requirements; besides, conventional text annotations are applied to use cases documentation for describing non-functional requirements
The resulting document is labelled as the current activity: *Domain Requirements Description* (or briefly DRD) document. As described on the ASPECS website [7] the DRD fragment is described (a more complete description is on the website) as follows:
Goal
The global objective of the Domain Requirements Description (DRD) activity is to provide an overview of the system's functionalities. This activity thus aims at gathering needs and expectations of application stakeholders and at providing a complete description of the behaviour of the application to be developed.

```
<profile:CurrencyService rdf:ID="CurrencyConverterProfile">
<service:isPresentedBy rdf:resource="#CurrencyConverterService"/>
<profile:serviceName xml:lang="en">Price Converter</profile:serviceName>
<profile:textDescription xml:lang="en">Converts the given price
  to another currency.
  </profile:textDescription>
<profile:hasInput rdf:resource="#InputPrice"/>
<profile:hasInput rdf:resource="#OutputCurrency"/>
<profile:hasOutput rdf:resource="#OutputPrice"/>
<:profile:CurrencyService>
```

Figure 2: Example of OWL-S profile: the CurrencyConverterProfile from

Input
Use cases are deduced from a set of text descriptions of the system usage scenarios (that are supposed to be an input of this design process), quality and procedure documents and are gradually refined with stakeholder interviews and meetings.
Output
UML use case diagram is the notation suggested for adoption in this activity. Requirements are described in terms of use case diagrams. The result of the Domain Requirements Description activity is a functional description of the system composed of a hierarchical series of use case diagrams.
MAS Metamodel Elements
This fragment defines the following metamodel elements: Functional Requirements, Non-Functional Requirements and relates the following metamodel elements (Functional Requirements, Non-Functional Requirements).

2.3 TROPOS

Tropos [2] is an agent-oriented methodology based upon the i^* framework [26]. The TROPOS methodology emphasises early requirements analysis, the phase that precedes the prescriptive requirements specification. The objective of the early requirements phase is to capture and analyse the goals of stakeholders. The concepts used to model early requirements are: actors (which can either be agents, roles or positions), goals and actor dependencies. Agents are then introduced very early in the analysis of a system as entities in charge of the realisation of goals by the dependencies relationships. The TROPOS methodology distinguishes two sorts of goals: soft-goals and hard-goals (also called goals). Softgoals represent vaguely defined goals for which there are no simple way to decide if it is fulfilled or not.
An example of early requirement analysis is given in figure 3. This simple example introduces two actors, namely the client who wants a product to be realised and a provider who will realise the product. The goal product realisation introduces a dependance of the client to the provider. We have also defined two soft-goals Increase benefits and Happy customer which make the provider dependant on the client.

This early requirement analysis enables to describe with a high level point of view the requirements of the model-to-be. This model is enriched by goal modelling. The goal modelling consists in the analysis of an actor by either means-end analysis, AND/OR decomposition or contribution analysis. This modelling is illustrated in figure 4. In this figure the provider actor is refined and some internal goals are elicited.

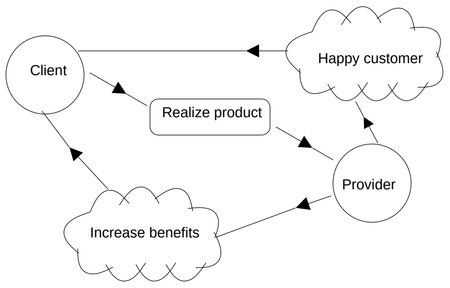


Figure 3: Early requirements for a collaborative design project

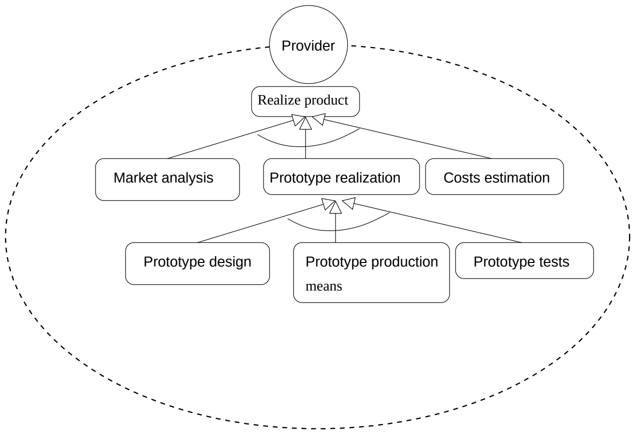


Figure 4: Goals analysis

The product realisation goal of actor provider is decomposed in three sub-goals by an AND decomposition. It means that in order to fulfil the Product realisation goal the three sub-goals need to be fulfilled. These three sub-goals are: Market analysis, Prototype realisation and Costs estimation. Each of these goals may in turn be refined and decomposed in sub-goals as illustrated for the Prototype realisation goal which can be decomposed, again with an AND decomposition, in three sub-goals. These sub-goals are: Prototype design, Prototype production means and Prototype tests. The goal modelling may continue until a satisfying level of detail is reached.

3. FRAGMENT SELECTION SCENARIO

This section illustrates our approach for fragments documentation and selection through a scenario. The next subsection gives some details about the scenario. The fragment documentation subsection explains how fragments can be represented as OWL-S services. Fragments are only described by OWL-S profiles in this scenario. The whole description of fragments by profile, process model and grounding would have taken too much space for presentation and discussion. Moreover, with profiles only one can use reasoning and matchmaking mechanisms as described in [16]. The scenario illustrates the reuse of existing metamodels underlying methodologies in the construction of a general ontology. Eventually, the selection subsection discuss how a fragment can be retrieved according to the method engineer aims.

3.1 Scenario description

The ASPECS process is dedicated to HMAS. One of the main principle underpinning the ASPECS process is the identification of several organisational levels from the study of the problem at hand. The resulting levels are described in the output of the Organisation Identification activity output, see figure 5. These organisational levels may give hints in order to design holarchies. Currently in ASPECS the identification can be done following two different strategies. The first strategy consists in using the structure of the use cases resulting of the Domain Requirement Description (DRD) activity which produces a use case diagram, see figure 5. More specifically, each use case is to be associated with an organisation in charge of it realisation. If a use case depends on another use case (includes relationship) then the associated organisations will be at different levels. The second strategy consists in using the ontology result of the Problem Ontology Description (POD) activity which produces an ontology as output, see figure 5. The proposed heuristic is to search for self-containing structures that may guide holarchies design. Obviously there are other ways to analyse a problem in order to find organisational levels that may give birth to holarchies.
A method engineer who is trying to find another strategy for holarchy design may search the method base by using the following request "which fragment(s) can support a requirement analysis associated to a requirement decomposition?".

3.2 Fragment documentation

The use of OWL-S for fragment documentation requires the existence of an ontology that is the conceptualisation of the concerned methodology. The design of such ontologies can be a difficult and time consuming task. However, many MAS methodologies relies upon metamodels describing the methodologies concepts and their relationships [1]. These metamodels constitute a solid foundation for ontologies design.
For our example scenario, we have used the ASPECS [8] and TROPOS [25] metamodels as basis for the conceptualisation of their respective ontologies. Extracts of these two ontologies are presented in figures 6 and 7.

There are several strategies in order to relate ontologies such as equivalence, translation or metaontologies. A generic metaontology should describe and relate concepts of the different methodologies between them. For requirements the

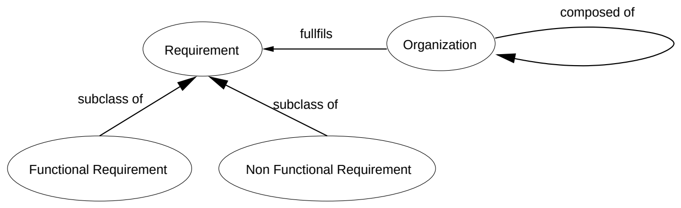


Figure 6: A portion of the ASPECS ontology dealing with requirements analysis

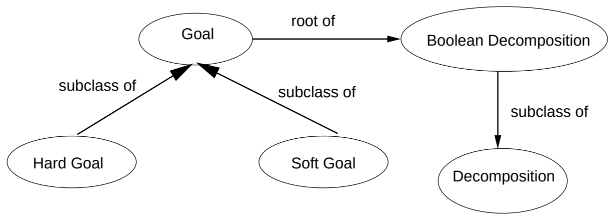


Figure 7: A portion of the TROPOS ontology dealing with goal analysis

ontology of figure 8, extracted from [10], is a conceptualisation of all requirements related concepts. The more generic concept is *AbstractRequirement*. The DRD ASPECS fragment has as outputs one or several Requirement (Functional or Quality) which is a subclass of *AbstractRequirement*. This is part of the ASPECS ontology, specifically the part related to the problem domain as stated in the ASPECS metamodel [8]. The TROPOS methodology, on the other hand, is based upon the concept of goal. No requirement concept (Functional or Quality) is present as goals and the analysis is goal-oriented. The metamodel describing the concepts of TROPOS [25] refines the concept of *Goal* in *HardGoal* and *SoftGoal*. Moreover, each *Goal* is associated to a *Decomposition*.
Clearly, the *Goal* concept of TROPOS is an answer to the method engineer request. Indeed, The concept of *Goal* is a subclass of *AbstractRequirement* as stated in the ontology figure 8 and the TROPOS ontology associate a *Decomposition* to *Goal*.
Among the fragments that compose the TROPOS methodology there is one that produces as output a goal diagram such as the one presented n figure 4 and that takes as input an actor diagram such as the one presented in figure 3. The OWL-S class of figure 9 defines the profile of the corresponding fragment. OWL-S profiles define inputs, outputs, results and preconditions. Here for the sake of clarity only inputs and outputs are presented. The class is a FragmentProfile which is a specific class created for fragments documentation and selection purpose and that inherits from Profile. The fragment, here viewed as a service, is named *TROPOS-EarlyRequirementAnalysis*. The output produced by this fragment is a TROPOS Goal Diagram (as the one presented in figure 4).

3.3 Fragment selection

The fragment selection in our approach relies on mechanisms that enable searches and reasoning with OWL and OWL-S. In this context there are several existing mechanisms such

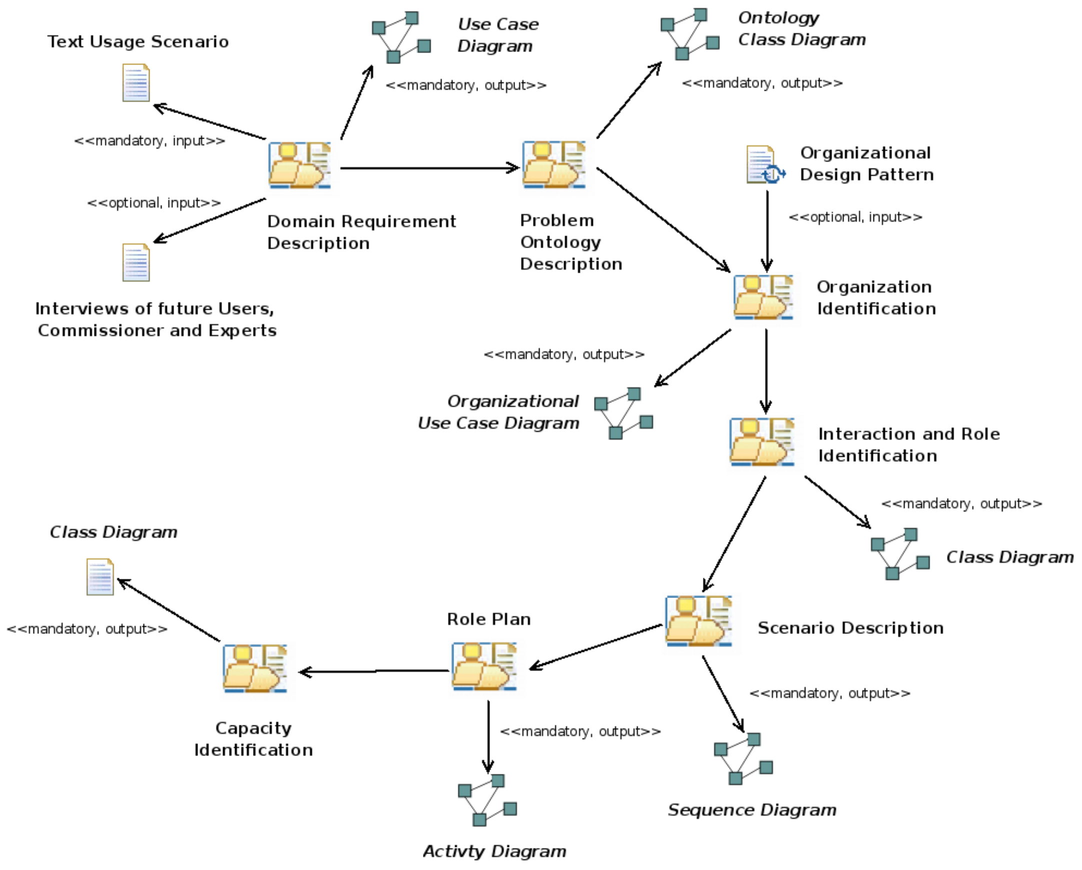


Figure 5: aspecs System Requirement Phase

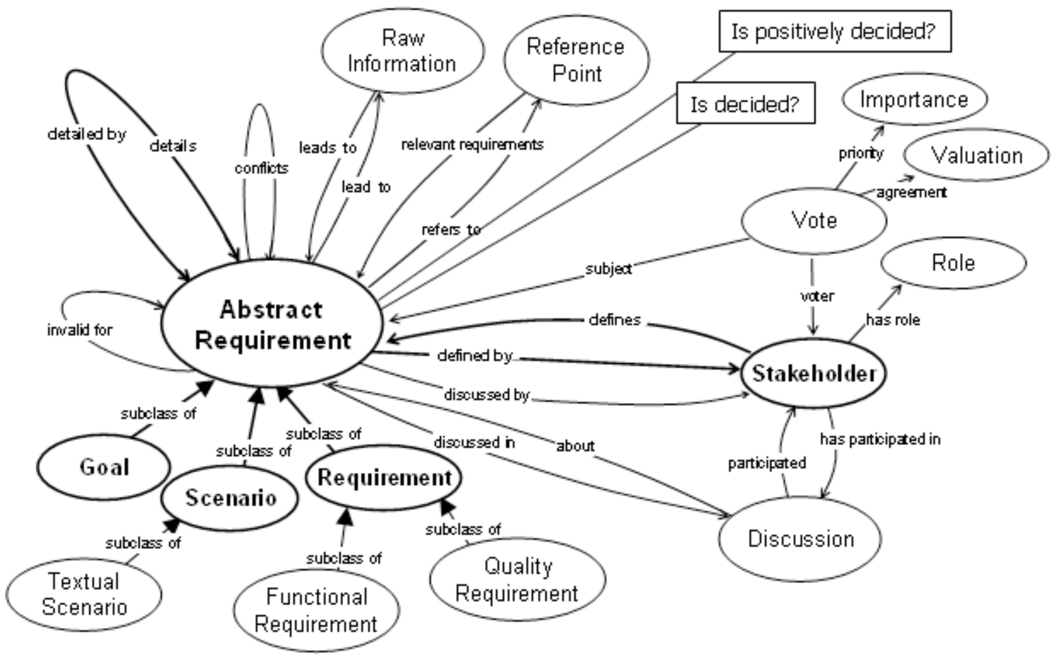


Figure 8: Requirements Ontology

```
<FragmentProfile rdf:ID="EarlyRequirementDescription">
   <serviceName>TROPOS-EarlyRequirementAnalysis</serviceName>
   <hasInput>
      <process:Input rdf:about="&TROPOS;#ActorDiagram">
         <process:parameterType rdf:datatype="&TROPOS;#TROPOSDiagram">
         </process:parameterType>
      </process:Input>
   </hasInput>
   <hasOutput>
      <process:Output rdf:about="&TROPOS;#GoalDiagram">
         <process:parameterType rdf:datatype="&TROPOS;#TROPOSDiagram">
         </process:parameterType>
      </process:Output>
   </hasOutput>
</FragmentProfile>
```

Figure 9: Example of TROPOS Fragment profile

as: SPARQL [18] which allow searches upon RDF triplets, whatever reasoning mechanisms based upon OWL-DL like Pellet [24] and the mechanisms applied to the discovery of service, see [16] for a survey. Up to now we have not implemented a software environment that help the method engineer to select appropriate fragments. The technique used in the presented scenario is to edit the ontologies wih Protg[11] and query them with the Pellet plugin for Protg.
The aim of the method engineer scenario was "which fragment(s) can support a requirement analysis associated to a requirement decomposition?". This question can be decomposed as (1) "which concepts, which are requirements, are associated to decomposition?" and consequently (2) "which fragments produce these concepts?". Using tools such as Protg [11] the method engineer can query the fragments ontology and obtain the answer *Goal* to the question (1), see figure 10. Now the question for the method engineer is which fragment produces the goal decomposition as an output? This question is answered by the fragment described in the preceding subsection see figure 9. This fragment takes a TROPOS Actor Diagram as an input. Since this input is not present in the ASPECS methodology the method engineer now needs a fragment that produces an Actor Diagram as an output. This fragment is present in the method bas and it is called *TROPOS-EarlyRequirementDescription*. This fragment input is a Problem Statement document that is also an input of the ASPECS methodology. The two identified fragments are sufficient for satisfying the goal pursued by the method engineer in the proposed scenario. The method engineer can then replace the fragments DRD and POD of the ASPECS methodology with the two TROPOS fragments in order to obtain a goal oriented analysis that enables the identification of holarchy levels.

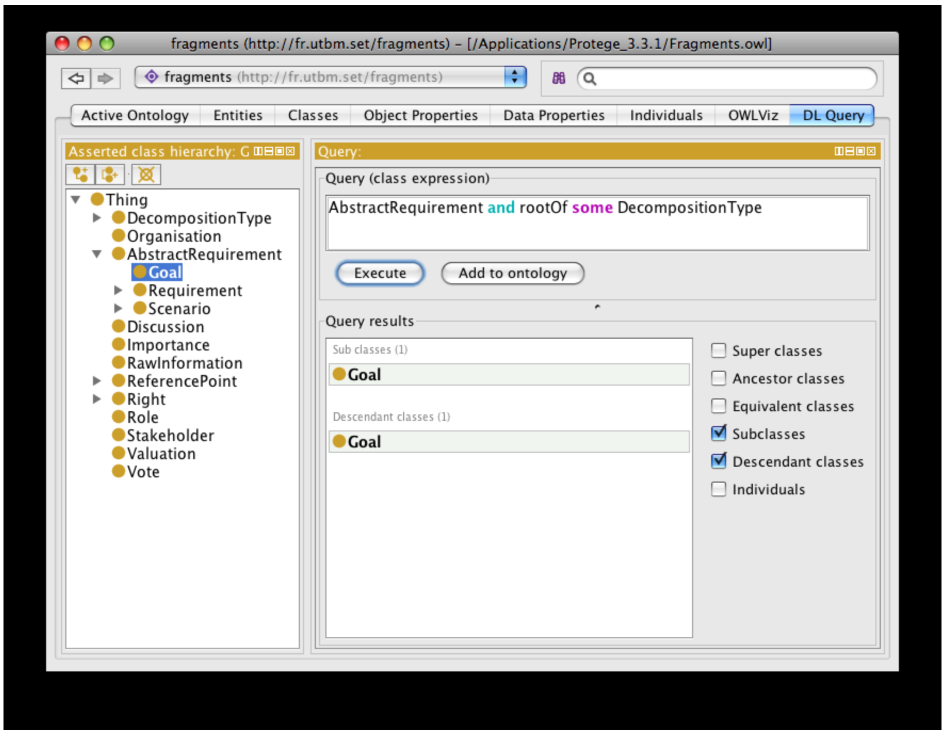
4. RELATED WORKS

Many researches has been done on Situational Method Engineering and method fragments. A few fragment bases already exist, for example OPF [9], which stands for OPEN Process Framework, is a website that includes many documented fragments. These fragments follow a process metamodelling approach describe [12]. Despite the fact that the work done is very important the documentation of fragments is text-based using natural language. This approach may be ambiguous and doesn't allow search and semantic annotations.The authors of [20] propose the use of metrics to compare fragments (named chunks in their approach). These metrics estimate semantic and structural similarities between concepts of different methodologies. This approach seems less intuitive than the use of ontologies which are designed to represent conceptualisations. Moreover, ontologies in general and OWL-S allow the use of powerful tools for searches and matchmakings.
Rules that describe how to assemble method fragments into a situational method are reported in [3]. These rules are defined in natural language and then formalised using predicate logic. The aim of these rules is to state the generic conditions that must be satisfied when a method engineer assemble fragments to create a new method, whatever the method. The semantic of each fragments is not clearly tackled and no search facilities are provided.
In [5] authors have made the hypothesis that a MAS metamodel can be used to guide the process composition activities . An algorithm is used to compute a list of MAS metamodel elements that allow to retrieve the method fragments. This approach doesn't tackle the semantic underlying the metamodel concepts but it is supported by a specific method base, as reported in [21]. The structure of this method base is based upon a method fragment definition that has been initially proposed within a FIPA Methodology Technical Committee and then refined by some of the authors of this paper [6]. This fragment definition will now considered for future improvements and standardization from a newborn IEEE FIPA working group on Design Process Documentation and Fragmentation [1].
A comparison of some approaches for method engineering can be found in [22].

5. CONCLUSION

In this paper an approach for method fragments documentation and selection from method base is proposed. This approach is based upon OWL-S and the view of fragments as services. Each fragment is described by means of OWL-S classes. This paper presents the use of the profile part of an OWL-S compliant documentation of a fragment-service through a scenario where a method engineer searches method

[1]See website at: http://www.fipa.org/subgroups/DPDF-WG.html



```
AbstractRequirement and rootOf some DecompositionType
```

☐ Super classes
☐ Ancestor classes
☐ Equivalent classes
☑ Subclasses
☑ Descendant classes
☐ Individuals

Figure 10: Example of reasoning performed with the support of Protg.

fragments that allow to obtain a requirements decomposition from problem statements. The aim of the method engineer is to enrich an existing methodology, namely ASPECS. The result of the search includes two TROPOS fragments that decompose problem statements using goals and goals decomposition. Moreover, matchmaking mechanisms and searches can produce fragments that do not answer exactly to the problem at hand but that can be adapted to fit that. The use of OWL-S description for method fragments combined with the ontological description of methodology concepts presents many advantages: *(i)* method engineers can annotate and give semantic to method fragments, *(ii)* fragments are precisely described, and *(iii)* searches and reasoning can be done upon them by using query or matchmaking mechanisms related to OWL and OWL-S. For instance, in the proposed example scenario, the concept *Abstract Requirement* that is a generalization *of Goal* and *Requirement,* allowed to abstract from two existing methodologies (ASPECS and TROPOS) using different requirements analysis techniques and to reuse their corresponding fragments.
The two unused parts of OWL-S, namely process and grounding, may also be of interest for method engineers. Indeed, the process part can describe the activities done within fragments and grounding may be useful in describing how to use fragments and how to adapt them.
The next step of this work consists in designing a software environment coupled with a fragment repository in order to help method engineers in fragments documentation and selection tasks. The use of MAS within such an environment will enable the integration of powerful existing tools that handle ontologies and their collaboration in the development of a complex software engineering tool.

6. REFERENCES

[1] BERNON, C., COSSENTINO, M., GLEIZES, M. P., TURCI, P., AND ZAMBONELLI, F. A study of some multi-agent meta-models. In *AOSE* (2004), J. Odell, P. Giorgini, and J. P. Müller, Eds., vol. 3382 of *Lecture Notes in Computer Science*, Springer, pp. 62–77.

[2] BRESCIANI, P., GIORGINI, P., GIUNCHIGLIA, F., MYLOPOULOS, J., AND PERINI., A. TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems 8*, 3 (May 2004), 203–236.

[3] BRINKKEMPER, S., SAEKI, M., AND HARMSEN, F. Assembly techniques for method engineering. *Lecture Notes in Computer Science 1413* (1998), 381–??

[4] COSSENTINO, M. From Requirements to Code with the PASSI Methodology. In *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. Idea Group Publishing, Hershey, PA, USA, 2005, ch. IV, pp. 79–106.

[5] COSSENTINO, M., GAGLIO, S., GALLAND, S., GAUD, N., HILAIRE, V., KOUKAM, A., AND SEIDITA, V. A mas metamodel-driven approach to process composition. In *AOSE'08* (2008).

[6] COSSENTINO, M., GAGLIO, S., GARRO, A., AND SEIDITA, V. Method fragments for agent design methodologies: from standardisation to research. *International Journal of Agent-Oriented Software Engineering (IJAOSE) 1*, 1 (2007), 91–121.

[7] Cossentino, M., Galland, S., Gaud, N., Hilaire, V., and Koukam, A. Aspecs methodology. http://aspecs.org.

[8] Cossentino, M., Gaud, N., Galland, S., Hilaire, V., and Koukam, A. A holonic metamodel for agent-oriented analysis and design. In *HoloMAS'07* (2007).

[9] Firesmith, D. Open process framework. http://www.opfro.org/f.

[10] Fritsch, D., Lehmann, J., Lauenroth, K., Lohmann, S., and Riechert, T. Requirements ontology. http://softwiki.de/RequirementsOntology.

[11] Gennari, H., J., Musen, A., M., Fergerson, W., R., Grosso, E., W., Crubezy, M., Eriksson, H., Noy, F., N., Tu, and W., S. The evolution of protege: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies 58*, 1 (2003), 89–123.

[12] Henderson-Sellers, B. Process metamodelling and process construction: Examples using the OPEN process framework (OPF). *Ann. Software Eng 14*, 1-4 (2002), 341–362.

[13] Hilaire, V., Koukam, A., Gruer, P., and Muller, J.-P. Formal specification and prototyping of multi-agent systems. In *ESAW* (2000), A. Omicini, R. Tolksdorf, and F. Zambonelli, Eds., no. 1972 in LNAI, Springer Verlag.

[14] IEEE, Ed. *Software Engineering Body of Knowledge.* IEEE Computer Society, 2004.

[15] Iglesias, C., Garijo, M., Gonzalez, J., and Velasco, J. *Analysis and design of multi-agent systems using MAS-CommonKADS*, vol. 1365 of *LNAI*. Springer-Verlag, 1998, ch. Intelligent agents IV: Agent theories, architectures, and languages, pp. 313–326.

[16] Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D. L., Sirin, E., and Srinivasan, N. Bringing semantics to web services with OWL-S. In *First International Workshop on Semantic Web Services and Web Process Composition* (2004).

[17] MINDSWAP. Owl-s services. http://www.mindswap.org/2004/owl-s/services.shtml.

[18] Prud'hommeaux, E., and Seaborne, A. SPARQL query language for RDF. W3C recommendation, W3C, Jan. 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[19] Ralyté, J., and Rolland, C. An approach for method reengineering. *Lecture Notes in Computer Science 2224* (2001), 471–??

[20] Ralyté, J., and Rolland, C. An assembly process model for method engineering. *Lecture Notes in Computer Science 2068* (2001), 267–??

[21] Seidita, V., Cossentino, M., and Gaglio, S. A repository of fragments for agent systems design. *Proc. Of the Workshop on Objects and Agents (WOA06)* (2006).

[22] Seidita, V., Ralyté, J., Henderson-Sellers, B., Cossentino, M., and Arni-Bloch, N. A comparison of deontic matrices, maps and activity diagrams for the construction of situational methods. In *CAiSE'07 Forum, Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering, Trondheim, Norway, 11-15 June 2007* (2007), J. Eder, S. L. Tomassen, A. L. Opdahl, and G. Sindre, Eds., vol. 247 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[23] Simon, H. A. *The Science of Artificial*, 3rd ed. MIT Press, Cambridge, Massachusetts, 1996.

[24] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. Pellet: A practical OWL-DL reasoner. *J. Web Sem 5*, 2 (2007), 51–53.

[25] Susi, A., Perini, A., Mylopoulos, J., and Giorgini, P. The tropos metamodel and its use. *Informatica (Slovenia) 29*, 4 (2005), 401–408.

[26] Yu, E. Towards modelling and reasoning support for early-phase requirements engineering. In *3rd IEEE Int. Symp. on Requirements Engineering* (1997), pp. 226–235.