MEMOS: A Methodology for Modeling Services

Mick Kerrigan
Semantic Technology Institute
University of Innsbruck
Austria
mick.kerrigan@sti2.at

Barry Norton
Institute AIFB
Karlsruhe Institute of Technology
Germany
barry.norton@kit.edu

Elena Simperl
Institute AIFB
Karlsruhe Institute of Technology
Germany
elena.simperl@kit.edu

ABSTRACT

The Semantic Business Process Management (SPBM) approach from the SUPER project utilizes a Semantic Execution Environment (SEE) for the automatic discovery, composition, mediation, and invocation of Web services. In order to enable the Semantic Execution Environment, an engineer must create semantic descriptions of functional, non-functional, and behavioural aspects of Web services and end-user requirements. In this paper we introduce MEMOS, a methodology for modeling services that provides a detailed description of the different activities, tasks, roles, and artifacts that exist within a Semantic Web Service (SWS) engineering project, from both a service provider and service requester perspective. By introducing the first methodology for Semantic Web Services, we aim to support engineers in their development projects, ultimately improving quality, reducing cost, and enabling the adoption of Semantic Web Services at large.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Interoperability;
D.2.10 [**Design**]: Methodologies

General Terms

Human Factors, Design

Keywords

Semantic Web Services, WSMO, Methodology

1. INTRODUCTION

Business Process Management (BPM) is an established discipline whereby the processes of a company are modelled, monitored, managed, and adapted according to business experts' viewpoint, well-separated from the lower-level IT concerns associated with their realisation. Meanwhile the approach of Service-Oriented Architecture (SOA) has made strides towards supporting the requirements of agile cross-organisational business processes at this implementation level. Semantic Business Process Management (SBPM) [9] is a recent approach based on the application of ontology-based semantics to bridge the gap between the business analyst's and IT department's viewpoints in BPM, which is an application of the principle of *ontological role separation*. Many SBPM approaches use Semantic Web Services (SWS) and semantics-driven execution.

Semantic Web Services (SWS) represent the next evolutionary step forward in Service Oriented Computing, namely the addition of ontology-based semantic descriptions for each service, comprising a formal description of the services functionality, non functional aspects, and external behaviour. Once services are described semantically, many of the tasks involved in using them can be automated and real dynamism can be added to applications using the Semantically-enabled SOA paradigm. SWS have reached a maturity level where there is a shared understanding within the research community, which is supported by conceptual models, languages, and tools and there is considerable interest in the use of SWS as a technology for enabling the dynamic discovery, composition, mediation, and invocation of Web services. However, the barrier to the adoption of this new technology is its complexity and a lack of understanding of which activities need to be performed in order to achieve successful results.

In order to achieve this meeting between different communities beyond the sphere of research it is necessary to concretely and methodically describe the results and shared understanding of 7 years research and development in SWS. In particular the conceptual models, languages, frameworks, and tools must be placed in the context of methodologies, best practices, and guidelines. The introduction of methodological support for SWS will enable engineers to receive the *consistency of action* [12] that a methodology provides, and improve the quality of the resulting descriptions in terms of meeting cost estimates, having all of the functionality promised, and delivering in the right time frame.

We begin this paper by giving a brief overview of SBPM and SWS (Section 2) and then summarize our preliminary analysis of how SWS are currently being used in the SBPM community from [11] (Section 3). In this analysis we identify a set of scenarios for SWS that describe the high-level tasks that must be performed by engineers in order to realise each of them, with these scenarios acting as a starting point for the methodology. We then introduce a Methodology for Modeling Services (MEMOS), which provides a systematic approach to the implementation of Semantic Web Services

SBPM '10 Crete, Greece

by defining the activities and tasks that must be performed by particular actors in each of the phases of the software development cycle (Section 4). This methodology has undergone evaluation in the form of both professional reviews and a case study (Section 5), which have highlighted open issues and motivate our future work (Section 6).

2. BACKGROUND

To support the separation of business analysts' and IT viewpoints of business processes the SUPER project has introduced two main ontologies: the *Business Process Modelling Ontology* (BPMO)[1] and a *Semantic BPEL* model, which is grounded for execution to a compliant extended WS-BPEL 2.0 schema, BPEL4SWS [17]. Common features between the two can be mediated using ontology-based rules [18].

BPMO's stated intention is to "[model] business processes at the semantic level, integrating knowledge about the organisational context, workflow activities and Semantic Web Services." The workflow aspect is conceptualised via the abstractions encoded in Workflow Patterns [21] and aims to be graphically represented in a fragment of the Business Process Modelling Notation (BPMN) [19]. The patterns covered are also intended to abstract over the features of Event-driven Process Chains (EPCs), the extended model of which in the ARIS toolset is inspiration for the modelling of organisational context [20].

The Web Service Modeling Ontology (WSMO) [8] is a conceptual model for Semantic Web Services and has four top level elements, namely Ontologies, Web Services, Goals and Mediators. Ontologies are the basis for the other descriptions by providing the terminology that they use. WSMO Web Services provide a semantic description of both the function of a service, in terms of a Capability, and the mechanism for interacting with it, in terms of an Interface. A WSMO goal allows for the requirements of the requester to be semantically described. Finally, WSMO Mediators provide a means to resolve heterogeneity issues that inevitably occur between the other elements due to the open and distributed nature of the Web.

WSMO's service model is primarily used in two regards in BPMO. The concept of goal allows the requirements for external tasks to be functionally specified, along with non-functional requirements related to, for instance, Quality of Service. The concept of mediator allows both the specification of necessary data mediation within a process, as well as a mediation process to be specified between a set of processes that are otherwise lacking in mutual conformance. In translation to Semantic BPEL mediators are intended to play the same role [16]. Goals, on the other hand, may either be left in place for run-time matching to a web service, as described below, or may be replaced with the semantic description of a suitable service in the executable process.

A number of other conceptual models and languages exist for semantically describing services, including OWL-S [14] and WSMO-Lite [22]. Recently the Reference Ontology for Semantic Service Oriented Architectures (SSOA-RO) [2] was defined in the OASIS Semantic Execution Environment technical committee (SEE-TC). It provides a unambiguous definition in RDF-S of the different concepts that exist within a SSOA inspired by existing models and languages. Transformations from this reference ontology to and from OWL-S, WSMO, and WSMO-Lite have also been defined in [6]. Thus in this paper we use the terminology from the SSOA-RO to define the MEMOS methodology such that it can be applied to WSMO as well as the other models and languages. The SSOA-RO terminology is introduced throughout the next section.

[1]Final SUPER version submitted to SBPM.
Alternate link via post-SUPER pre-standardisation activity in STI Conceptual Models for Services Working Group:
http://cms-wg.sti2.org/reports/

3. ENGINEERING SCENARIOS FOR SEMANTIC WEB SERVICES

In [11], we performed an analysis of recent projects on the topics of Semantic Web Services and Semantic Business Process Management (SBPM), as well as a survey of SWS experts, in order to understand how Semantic Web Services are being used in the community. From this analysis, in [11] we also defined a set of engineering scenarios that describe the high-level tasks that must be performed by engineers in order to realise each of them. The benefit of these scenarios is they can be combined by an engineer in order to design rich and complex systems, while still allowing the engineer to have a clear understanding of the SWS artifacts that must be implemented in order to enable the final system. The scenarios are a starting point for the MEMOS methodology and are summarized here in terms of the SSOA-RO, a full description of which can be found in [4]:

Scenario 1: Using the SSOA-RO for Service Advertisement: A discovery broker service can be used to find service descriptions advertised by providers based on a requesters goal description. There are two common approaches to service discovery, described in the scenarios below. If service ranking is required in either scenario, providers should annotate their service descriptions with non functional properties describing the quality of service aspects of their Web service and requesters should provide their preferences over non functional properties in their goal descriptions.

Scenario 1a: Capability-based Service Advertisement: In this scenario the provider creates a functional description of the service to be advertised in the form of a capability description. Similarly, the service requester provides a capability description of their functional need. Matches are found by comparing the requesters capability description against provider capability descriptions.

Scenario 1b: Mediator-based Service Advertisement: In this scenario mediators are used to define matches between service descriptions and goal descriptions. These matches are defined at design time and thus the process of discovery is a simple lookup of a mediator. Mediator-based discovery is especially suitable in an environment where the number of services is limited.

Scenario 2: Using the SSOA-RO for Service Invocation: In a different context the services needed within a system may already be known, so no discovery is necessary; however automatic invocation of these services may be needed, particularly in cases where service interfaces change regularly. There are two ways in which service invocation can occur:

Scenario 2a: Service Choreography-based Invocation: The provider creates a process model, entitled a chore-

ography, that describes the external behavior of their service, i.e. how the requester should interact with the service. The choreography is accompanied by a grounding that enables ontological instances to be sent to the service according to a particular data schema. The requester need only provide the relevant ontological instances to invoke the service.
Scenario 2b: Goal Choreography-based Invocation: Scenario 2a is only possible in cases where all the information needed to execute the service choreography is available prior to the execution. If new data needs to be generated based on the responses from the service, then the requester also requires a choreography such that a conversation between the requester and provider can be made.

Scenario 3: Using the SSOA-RO for Service Composition: Scenario 1 enables the automatic discovery of services; however, it is often the case that no single service can fulfil a requesters goal description. In such a case it may be possible to combine a number of services in order to fulfil the requesters requirements. There are two approaches to service composition:
Scenario 3a: Design-time Service Composition: Some actor manually creates a new service description containing an orchestration, which brings together individual service and goal descriptions to deliver a composite functionality. This scenario can be combined with scenarios 1 or 2 so that this service description can be automatically discovered or invoked.
Scenario 3b: Run-time Service Composition: An orchestration of service descriptions is created automatically to fulfil a goal description. The composition process is end-user-guided, through the specification of a capability description, non functional preferences, a target choreography, or a partial orchestration.

Scenario 4: Engineering Ontologies in the SSOA-RO Context: Scenarios 1, 2, and 3 require ontologies to enable service and goal descriptions to be created. While engineers should use existing ontology engineering methodologies, the following scenarios should be considered in the ontology engineering process:
Scenario 4a: Engineering Ontologies from a Data Schema: Using the provider or requesters data schema, e.g. the XML Schema of a SOAP Web service, as input to the ontology engineering process makes it easier to create a grounding for the resulting ontology; however no shared understanding with other parties exists and heterogeneity issues must be resolved later.
Scenario 4b: Reusing Existing Ontologies: Reusing existing ontologies to describe service and goal descriptions results in few heterogeneity issues between requesters and providers, but the process of creating a grounding becomes more complicated due to the potential gap between the data schema used by the provider or requester, and the reused ontologies.
Scenario 4c: Reengineering Existing Ontologies: Existing ontologies are reengineered taking into account the provider or requester data schema. Creating a grounding is more complex than in 4a and more heterogeneity exists than in 4b, but better than the worst case in both.

Scenario 5: Enabling Interoperation Between Ontologies: Using scenarios 4a and 4c results in ontologies that are locally relevant but not shared with others in the community. Discovery, invocation, and composition in scenarios 1, 2, and 3 will not function correctly unless the ontologies used by requesters and providers are aligned. Therefore, to enable interoperability between requester goal descriptions and provider service descriptions it is necessary to define an ontology to ontology mediator (ooMediator), between the ontologies that they use. An ooMediator is usually accompanied by a mapping document containing mappings between the different elements in the source and target ontologies.

4. MEMOS: A METHODOLOGY FOR MODELING SERVICES

The Methodology for Modeling Services (MEMOS) provides a systematic approach to the implementation of Semantic Web Services using the SSOA-RO by defining the specific activities and tasks that must be performed by particular actors in each of the phases of the Software Development Cycle (SDC). The methodology is designed considering the different scenarios in which Semantic Web Services are currently being used by the community, as described in Section 3. The MEMOS activities and tasks are defined in a Software Development Process Model (SDPM) independent way such that they can be combined into a process with activities and tasks from other methodologies, for example the OASIS FWSI Web Service Implementation Methodology [3].

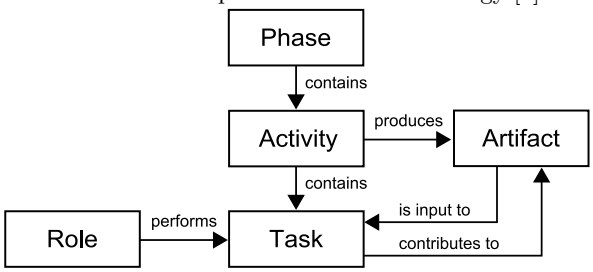

Figure 1: Relationship between Phases, Activities, Tasks, Roles and Artifacts in MEMOS

MEMOS is structured around the Software Development Cycle and defines activities that should be conducted in the context of the requirements, design, implementation, testing, and installation & checkout phases of the Software Life Cycle (SLC) as defined by the IEEE in [1]. Activities in the MEMOS methodology define a collection of common tasks that lead towards the output of a particular artifact. In certain activities, particularly in the requirements phase, tasks within a particular activity are split into provider tasks and requester tasks. Provider tasks are undertaken by service providers as they attempt to expose their services according to the scenarios defined in section 3. Requester tasks are similarly performed by those attempting to use Semantic Web Services to fulfil some functional need within an application. A task in the MEMOS methodology is a unit of work that contributes to the completion of a given activity and its output artifact(s). One or more roles perform the task by utilizing the provided input artifacts, according to the specified guidelines, to produce the output artifacts. Figure 1 illustrates the relationship between phases, activities, tasks, roles, and artifacts, which is in-line with the IEEE standard documentation.

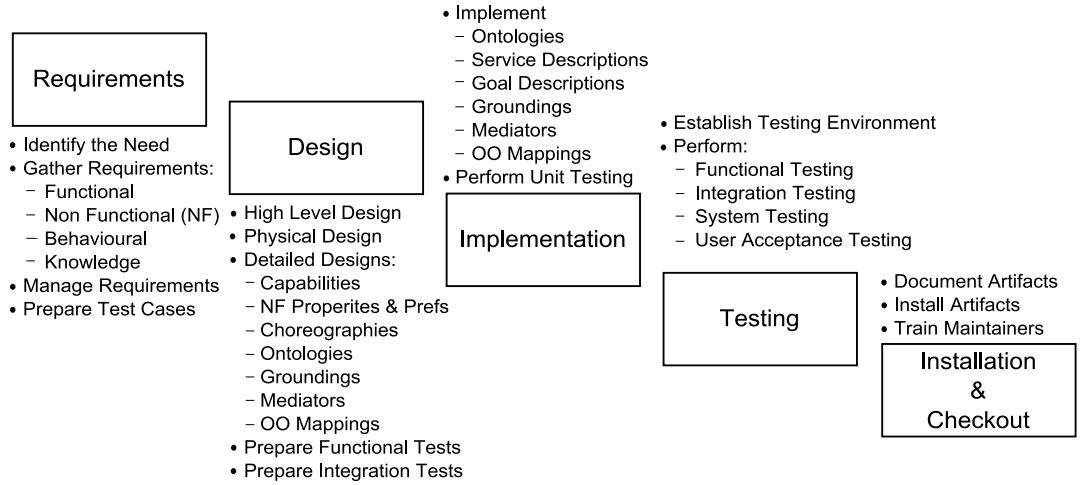

Figure 2: Overview of Activities in the MEMOS Methodology

The methodology aims to support SWS engineers and aid them to realize system that use scenarios 1, 2, 4, and 5 as described in Section 3. The pace at which the different SWS artifacts have been adopted and used to date in the community varies, and this is especially true for orchestrations. Thus service composition (scenario 3) is deemed out of scope for this version of the methodology and will be added later when the community has more experience. It should also be noted that a large amount of effort has been spent within the ontology engineering community developing methodologies for creating, reusing, and reengineering ontologies. While the MEMOS methodology has tasks in each of the relevant software development cycle phases related to the development of ontologies, these tasks are delegated to an appropriate ontology engineering methodology, which is selected during the requirements gathering phase.

MEMOS is made up of 28 activities, which are in turn made up of 94 individual tasks with accompanying guidelines, an overview of which can be seen in Figure 2. The tasks are performed by 14 roles including usual software engineering roles, for example *requirements analyst*, *designer*, and *domain expert*, and those specific to SWS development, namely *ontology engineer*, *semantic service engineer*, and *mapping engineer*. Due to space restrictions it is not possible to describe all 94 MEMOS tasks in detail, thus in the following sections we provide an overview of each of the MEMOS activities and summarize the tasks within them. A full description of all activities and tasks can be found in [4].

4.1 Requirements Phase

The requirements phase is made up of 7 activities and aims to solicit requirements from stakeholders and domain experts regarding the problem to be solved:

Activity R1 - Identify the Need: The aim of this activity is to discover the exact problem that needs to be solved by the development project. The requirements analyst begins by identifying the stakeholder and the relevant domain experts needed in this process and goes on to elicit a problem statement from these individuals. Once a problem statement exists the requirements analyst should identify those scenarios from section 3 needed to solve the problem and finally gather the knowledge sources from the domain expert needed in the requirements phase, for example service or application design documentation, WSDL documents, and XML Schemas.

Activity R2 - Gather Functional Requirements: If scenario 1 is identified during activity R1 then the requirements analyst should gather requirements related to the functional aspects of the service description or goal description to be created. In a provider context this involves the requirements analyst understanding the functionality offered by the Web service, the specific functionality that the stakeholder wants to advertise, and the discovery broker services where the advertisement will be made. In a requester context the requirements analyst should understand the application in question, determine the functional need that this application has, and the discovery broker services where the stakeholder wants to search for this functionality.

Activity R3 - Gather Non Functional Requirements: If scenario 1 is identified during activity R1 then the requirements analyst should gather requirements related to the non functional aspects of the service description or the goal description to be created. In a provider context this means understanding the non functional behaviour of the Web service to be described and choosing which of these aspects should be advertised, e.g. availability, security, obligations. In a requester context the requirements analyst should gain an understanding of the non functional aspects which are important to the application that will use services, and identify the importance that the stakeholder assigns to these non functional aspects.

Activity R4 - Gather Behavioral Requirements: If scenario 2 was identified during activity R1 then the requirements analyst should gather requirements related to the behavioral aspects of the service description or goal description to be created. In a provider context this means understanding the behaviour of the different operations on the Web service, the relationship between them, and the groupings of operations that should be advertised as choreographies. While in a requester context this means identifying the data that is available within the application which could be sent to services, and the data that would be expected as a result of invocation.

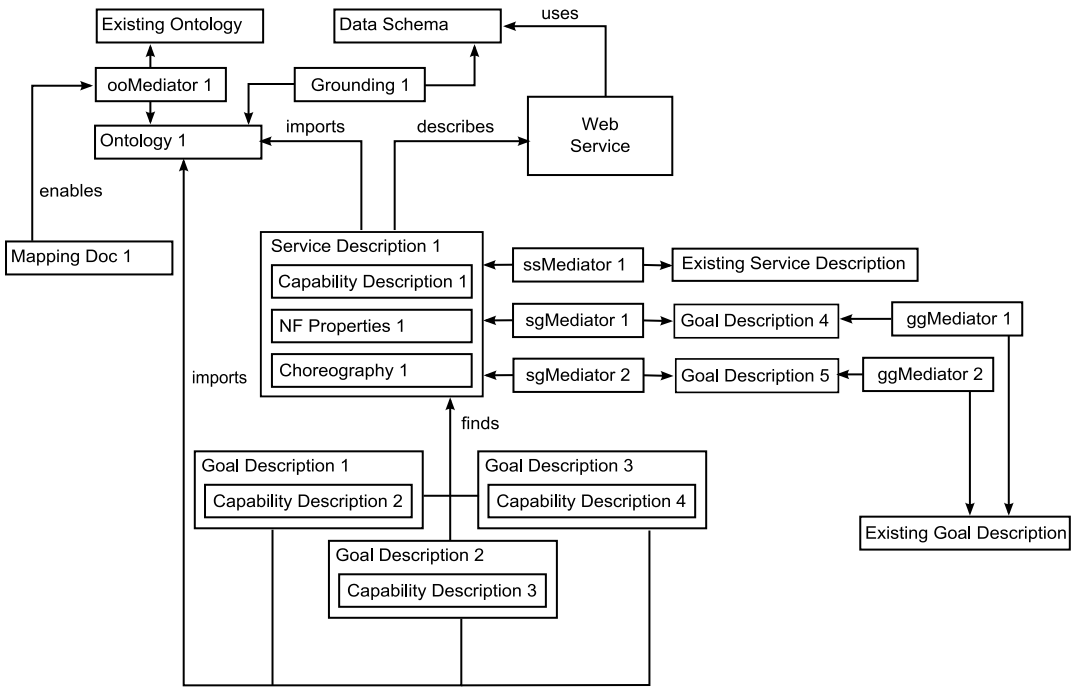

Figure 3: Example of a Providers High Level Architecture Diagram

Activity R5 - Gather Knowledge Requirements: Using the functional, non functional, and behavioral requirements as input, the requirements analyst can begin the process of requirements gathering for the ontologies that will be required to describe the functional, non functional, and behavioral aspects of the service description or goal description to be created. This activity involves the selection of an appropriate ontology engineering methodology and performing the activities from this methodology. The requirements analyst should also work with domain experts to identify those models that are used (or could be used) by potential collaborators and competitors.

Activity R6 - Manage Requirements: In this activity the requirements are checked for consistency and to ensure that they meet the original problem statement from the stakeholder. This activity also includes the identification of dependencies between requirements and the assignment of requirement priorities.

Activity R7 - Prepare User Acceptance and System Tests: The aim of this activity is to identify relevant test cases that can be used for system and user acceptance testing in the test phase of the development process. This activity also ensures that all of the requirements identified in the requirements specification are covered by test cases by creating a validation matrix.

4.2 Design Phase

The design phase is made up of 10 activities, and aims to build a well-organized representation of the set of artifacts needed to meet the gathered requirements.

Activity D1 - Define High Level Architecture: The aim of this activity is to define the overview architecture of all the service descriptions and goal descriptions needed to realize the gathered requirements, along with the ontologies that they use. The high-level architecture also contains the mediators that link service descriptions, goal descriptions and ontologies together, and where necessary the mappings that enable these mediators. Finally the high level architecture shows the different groundings that are required to enable interoperability between the SSOA-RO artifacts and the concrete services or applications they describe. An example of a providers high level architecture diagram can be seen in Figure 3, a requesters high level architecture diagram would look similar, except that it would be centred around the application with a functional need. An in depth explanation of Figure 3 can be found in [4].

Activity D2 - Define Physical Architecture: In this activity the architect and deployer work together to tie the high level architecture to a physical architecture. The physical architecture specifies the different repositories and broker services where the artifacts in the high level architecture will be registered and used. This activity also produces a step-by-step deployment plan, which will be followed in the installation and checkout phase.

Activity D3 - Design Capability Description: In this activity the designer should create a detailed design description for each of the capability descriptions defined in the high level architecture. This activity involves creating natural language descriptions of each of the preconditions, postconditions, assumptions, and effects that describe the functionality of the service. The designer should consider the information gathered about the discovery broker service where the capability will be used, as it may ignore service preconditions, or mandate the specification of a minimum number of postconditions or effects, etc.

Activity D4 - Design Non Functional Properties and Preferences: In this activity the designer should create a detailed design of the non-functional properties on each ser-

vice description and the non functional preferences of each goal description in the high level architecture. This activity involves taking the properties or preferences identified in the requirements phase and assigning values to each. In the case of preferences, this activity also involves assigning weights to each of the preferences according to the requesters non functional needs.

Activity D5 - Design Choreography: In this activity the designer should create a detailed design for the choreographies of each of the service descriptions and goal descriptions in the high level architecture. The main output of this activity is a dependency diagram, which captures the different operations on the service, their inputs, their outputs, and the dependencies that exist between them. Choreographies on goal descriptions have a similar diagram except that they capture the requested interface of a Web service rather than that of a real Web service.

Activity D6 - Design Ontology: In this activity the designer is responsible for designing each of the ontologies in the high level architecture according to the ontology engineering methodology selected in the requirements phase. The main job of the engineer in this task is to design the concepts, attributes, relations, axioms, and instances that make up the ontology. The designer should take note of the logical formalism identified in the high level architecture for this ontology, as the choice of formalism will have an impact on the design of the ontology, and, a particular ontology may need to be created in more than one formalism in order to support different broker services.

Activity D7 - Design Grounding: In this activity the designer should create a design for each of the groundings identified in the high level architecture. The dependency graph created in activity D6 contains a clear specification of the different schema elements that need to be sent and received via the choreography and can be used by the designer to identify those parts of the underlying schema that need to be transformable to and from ontologies. The designer should go on to identify the equivalent ontology elements for each of these schema elements.

Activity D8 - Design Mediators: In this activity the designer should create a list of the sources and targets for each mediator in the high level architecture.

Activity D9 - Design OO Mappings: In this activity the designer should create a design for each of the mapping documents identified in the high level architecture. This activity begins by identifying the different elements of the source ontology that need to be transformed to the target ontology, utilizing the functional, non functional, and behavioural requirements. The designer should go on to identify discrepancies between source and target data values and design a set of data value transformation services to resolve these discrepancies, e.g. if the values "John" and "Smith" from the source become the value "John Smith" in the target, a service which concatenates the values together would be required.

Activities D3 to D9 also involve the definition of functional tests for each of the detailed designs created. For example, in the context of capability descriptions this means identifying the requesters capability descriptions that should find a particular providers capability description, or in the context of OO Mappings the test designer should create pairs of ontology instances from the source and target ontologies, which are equivalent with one another.

Activity D10 - Prepare Integration Tests: While the test cases used for system tests, which were created in activity R7, look at the end-to-end behavior of the artifacts in the system, the test designer should create integration test cases that should focus on how two or more individual artifacts interact.

4.3 Implementation Phase

In the implementation phase the detailed design of the individual components within the high level architectural design are reduced down to concrete SSOA-RO artifacts. The implementation phase is made up of 7 activities as follows:

Activity I1 - Implement Ontology: In this activity the ontology engineer creates an ontology for each of the ontologies in the high-level architecture according to their respective detailed design specifications, by performing the implementation tasks from the chosen ontology engineering methodology.

Activity I2 - Implement Service Description & **Activity I3 - Implement Goal Description:** These activities involve the reduction of the capability description design, non functional design, and choreography design down into concrete implementations for each service description and goal description in the high level architecture respectively.

Activity I4 - Implement Grounding: In this activity, lifting and lowering mappings are created to realize each of the groundings in the high level architecture, according to the approach used by the targeted broker service. For example, lifting and lowering broker services exist that are powered by rules [13], XSLT mappings [7], while others exist that require a specific Java class to be written [7].

Activity I5 - Implement Mediator: This is a relatively trivial activity, with each of the mediators specified in the high-level architecture being created and linking to the respective source and target SSOA-RO artifacts.

Activity I6 - Implement OO Mappings: In this activity each of the mapping documents in the high level architecture are created according to the OO mappings design created in activity D9. [15] defines two approaches to creating mapping documents. In the *bottom-up approach*, the engineer starts by creating mappings between the most primitive ontological concepts in the source and target ontologies, which act as a minimal agreement between these ontologies upon which more complex mappings can be made. Intuitively the *top-down approach* starts with the engineer creating mappings for the more complex ontological concepts first and then drilling down to map the less complex ontological concepts as necessary.

Activity I7 - Unit Test: The purpose of this activity is to test particular units of functionality in the system of SSOA-RO artifacts. The semantic service engineer should execute the functional tests defined in activities D3 to D9 in this activity against the artifacts created in I1 to I6.

4.4 Testing Phase

The testing phase ensures that the software designed in the design phase and implemented in the implementation phase meets the requirements laid down in the requirements phase. In the MEMOS methodology it has just a single activity, as much of the work related to testing is performed across the other phases of the development cycle:

Activity T1 - Execute Test Plan: Unit testing has been

performed in the implementation phase to ensure that the created artifacts function as expected within the development environment. The testing of the artifacts is brought a step further in this activity, with the functional, integration, system, and user acceptance test cases, defined throughout the development process, being executed in a testing environment that is equivalent to where they will be deployed.

4.5 Installation & Checkout Phase

The installation & checkout phase finalizes the Software Development Cycle and comprises 3 activities, which ensure that the artifacts created in the implementation phase are documented, deployed to the locations where they will be available to end users, and that maintainers are trained in how to maintain these artifacts:

Activity IC1 - Document Artifacts: The aim of this activity is to create the documentation needed for end users and maintainers to successfully work with the SSOA-RO artifacts created in the project, including annotations on the artifacts themselves and printed and on-line materials, e.g. user manuals.
Activity IC2 - Install Artifacts: This activity involves the execution of the deployment plan created in D2 such that the created SSOA-RO artifacts are deployed to the locations where they will be available to end users. Crucially this activity also involves the re-execution of the test plan to ensure these artifacts function as expected in the deployment environment.
Activity IC3 - Train Maintainers: The aim of this activity is to train the maintainers, who must maintain the SSOA-RO artifacts in the deployment environment, such that they have the relevant knowledge to perform their jobs.

5. EVALUATION

A two step approach to the evaluation of the methodology has been taken, using professional reviews and a case study:

OASIS SEE-TC Professional Review: As we utilized the SSOA-RO as the main language for describing the artifacts in the MEMOS methodology, the first logical place to conduct professional reviews was within the OASIS SEE-TC that produced this specification. The MEMOS methodology is currently available as a working draft [4] within the SEE-TC and has received useful feedback from the members of this technical committee, who are all experts in the field of Semantic Web Services. This feedback has enabled the improvement of activities and tasks in the methodology, and their associated guidelines. In the coming months the MEMOS methodology working draft will be voted on by SEE-TC members in order to create a committee draft, which will act as a supporting document to the SSOA-RO specification.

SHAPE Professional Review: Independently the SHAPE European framework project[2], which develops the needed infrastructure and technology for using the new OMG Service Oriented Architecture Markup Language (SoaML) standard [5], has adopted the MEMOS methodology as their methodology for engineering SWS in the SoaML context. By integrating the activities and tasks from MEMOS into the SHAPE methodology, the engineers have shown that MEMOS is generally applicable to service modeling and can be easily adapted to new conceptual models and languages, in this case SoaML. Feedback from these engineers regarding the methodology has been very positive and their suggestions have been used to improve the guidelines associated with MEMOS tasks. Future case study activities plan in the SHAPE project will act as further evaluation of the application of the MEMOS methodology in the SoaML context.

Case Study: A first case study was recently conducted with a group of three Semantic Web Service experts, who had all developed Semantic Web Services before, and who are experts in the areas of functional descriptions, non functional descriptions, and behavioural descriptions respectively. The tourism domain was selected for the use case, and the experts were asked to act on behalf of a service provider to advertise a hotel and car booking Web service using the MEMOS Methodology. The case study was conducted over the course of a week, with the experts using WSMO as the conceptual model for developing the service descriptions, and completing an experience report for each of the activities and tasks in the methodology as they performed them. The experts were given a requirements and design document for the Web service they needed to describe, along with the WSDL description of that Web service, and access to the domain expert who developed it. The experts felt that the activities and tasks in the requirements phase were easy to conduct and that the resulting requirements specification contained all the information needed to perform the rest of the development project. Where they experienced difficulty was in the design of the high level architecture, due to its size and complexity. They suggested that this issue could be resolved through the addition of tools to support this activity. The implementation tasks were relatively trivial for them to perform given their previous experience, however they stated that the guidelines provided with the implementation activities are a useful resource for those with less experience. The experts used the Web Service Modeling Toolkit (WSMT) [10], an integrated development environment for Semantic Web Services through the WSMO paradigm, throughout the implementation and installation & checkout phases. The saw the availability of these tools as crucial to the successful completion of the activities in these phases. Overall, the participants clearly stated that the MEMOS methodology was useful, could be feasibly applied within a development project, and that they believed that the availability of such a methodology would improve the consistency and repeatability of development of Semantic Web Services. The experience reports generated by the experts led to the addition of a number of new tasks to the methodology, as well as the improvement of the guidelines for particular tasks.

6. CONCLUSIONS

In this paper we have given a broad overview of a Methodology for Modeling Services (MEMOS), which defines the activities, tasks, artifacts, and roles that exist across the different phases of the Software Development Cycle for Semantic Web Services. In all there are 28 activities and 94 tasks in the methodology performed by 14 roles. Each of the tasks is accompanied with detailed guidelines on how to achieve the best result when performing them. We direct readers wishing to go into the details of these guidelines to

[2]http://www.shape-project.eu

[4]. It should also be noted that the methodology is supported by development tools in the form of the Web Service Modeling Toolkit (WSMT) [10], an integrated development environment for modeling services semantically.

In terms of next steps, additional case studies will be conducted to further test the feasibility, usability, and usefulness of the methodology, and to further hone the guidelines and recommendations that accompany it. The methodology currently supports scenarios 1a, 1b, 2a, 2b, 4a, 4b, 4c, and 5 as defined in Section 3. In coming versions of the methodology, we will examine the use of orchestrations within the community for enabling service compositions and extend the methodology with support for creating these orchestrations, both at design-time (scenario 3a) and in runtime (scenario 3b). Finally, while the WSMT supports many of the activities and tasks that must be conducted within the methodology, it does not currently guide the roles through the methodology itself. Subsequent versions of the WSMT will be extended in this direction, such that the process of following the methodology can be made easier.

Acknowledgements

The work is funded by the European Commission under the projects ACTIVE, COIN, LarKC, MUSING, Service Web 3.0, SHAPE, and SOA4ALL.

7. REFERENCES

[1] Glossary of Software Engineering Terminology (IEEE Standard 610.12-1990), 1990. Available from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159342.

[2] OASIS Committee Draft 1, Reference Ontology for Semantic Service Oriented Architecture Version 1.0. November 2008. Available from http://docs.oasis-open.org/semantic-ex/ro-soa/v1.0/see-rosoa-v1.0.pdf.

[3] OASIS Committee Draft 1, Web Service Implementation Methodology. 2005. Available from http://www.oasis-open.org/committees/documents.php?wg_abbrev=fwsi.

[4] OASIS Working Draft, Modeling Services with the Reference Ontology for Semantic Service Oriented Architecture 0.1. November 2009. Available from http://www.oasis-open.org/committees/document.php?document_id=35702.

[5] Service Oriented Architecture Modeling Language (SoaML), Object Management Group. November 2008. Available from http://www.omgwiki.org/SoaML/.

[6] L. Cabral, M. Kerrigan, and B. Norton. D14.1 Evaluation Design and Collection of Test Data for Semantic Web Service Tools. Semantic Evaluation At Large Scale (SEALS) framework project (FP7-238975), 2009.

[7] F. Facca, S. Komazec, and I. Toma. WSMX 1.0: A Further Step toward a Complete Semantic Execution Environment. In *Proc. of the 6th European Semantic Web Conf. (ESWC 2009)*, Jun 2009.

[8] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services – The Web Service Modeling Ontology*. Springer, 2006.

[9] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel. Semantic business process management: A vision towards using semantic web services for business process management. In *Proceedings of IEEE Intl. Conf. on e-Business Engineering (ICEBE 2005)*, pages 535–540. IEEE Computer Society, 2005.

[10] M. Kerrigan, A. Mocan, M. Tanler, and D. Fensel. The Web Service Modeling Toolkit - An Integrated Development Environment for Semantic Web Services. In *Proc. of the 4th European Semantic Web Conf. (ESWC2007)*, June 2007.

[11] M. Kerrigan, B. Norton, E. Simperl, and D. Fensel. Semantic Web Service Engineering for Semantic Business Process Management. In *Proc. of the 4th Intl. workshop on Semantic Business Process Management (SBPM)*, June 2009.

[12] H. Kerzner. *Strategic Planning for Project Management using a Project Management Maturity Model*. John Wiley & Sons, 2001.

[13] D. Lambert and J. Domingue. Grounding Semantic Web Services with Rules. In *Proc. of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP2008)*, Dec 2008.

[14] D. Martin. OWL-S: Semantic Markup for Web Services. Member Submission 22 November 2004, W3C, Available from http://www.w3.org/Submission/OWL-S/.

[15] A. Mocan and E. Cimpian. Mappings Creation Using a View Based Approach. In *Proc. of the 1st Intl. Workshop on Mediation in Semantic Web Services (Mediate-2005)*, Dec 2005.

[16] J. Nitzsche and B. Norton. Ontology-based data mediation in BPEL (for semantic web services). LNCS. Springer, 2008.

[17] J. Nitzsche, T. van Lessen, D. Karastoyanova, and F. Leymann. BPEL for semantic web services (BPEL4SWS). In *Proceedings of 3rd International Workshop on Agents and Web Services in Distributed Environments (AWeSome'07)*, volume 4806 of *LNCS*. Springer, 2007.

[18] B. Norton, L. Cabral, and J. Nitzsche. Ontology-based translation of business process models. In *Proceedings of 4th International Conference on Internet and Web Applications and Services (ICIW 2009, to appear)*. IEEE Computer Society, 2009.

[19] Object Management Group. Business process modelling notation (BPMN) specification. Technical report, Object Management Group, 2006. http://www.omg.org/docs/dtc/06-02-01.pdf.

[20] A. Scheer, T. Oliver, and A. Otmar. Process modelling using event-driven process chains. In M. Dumas, W. van der Aalst, and A. H. M. ter Hofstede, editors, *Process-Aware Information Systems*, chapter 5, pages 117–166. Wiley, 2005.

[21] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(3):5–51, July 2003.

[22] T. Vitvar, J. Kopecky, J. Viskova, and D. Fensel. WSMO-Lite Annotations for Web Services. In *Proc. of the 5th European Semantic Web Conf. 2008 (ESWC 2008)*, June 2008.