ASG Development Process – Application and Service Engineering
Deliverable D6.III-2

Authors:
Theresa Lehner,
Joachim Bayer
Fabio Bella,
Alexis Ocampo

Funded and supported by the European Union Project ASG (FP6-IST-004617).

IESE-Report No. 134.06/E
Version 1.0
June 29, 2006

A publication by Fraunhofer IESE

Abstract

This document presents the processes for engineering services that can be developed on an ASG platform and applications using the ASG platform. The processes are tailored to different project contexts. To this end, we first describe a general process for the two engineering processes, application engineering and service engineering, and develop a number of scenarios that cover the different situations that can arise when an ASG platform is used. The scenarios provide customizations to the general process, so that the process fits optimally to a given project. The scenarios vary in different characteristics:

- Does the present domain ontology cover the semantic information about the service or the application, respectively
- Are new services developed to support an application or are the required services present already
- Are existing services or applications reused

We describe the different scenarios that result from combining these characteristics and present the respective process customizations

Keywords: ASG, service engineering, application engineering:

Acknowledgements

We would like to thank Andrea Zisman and Michel Lacroix for their valuable feedback, which helped us to improve this report and the described processes.

Executive Summary

The goal of Adaptive Services Grid (ASG) is to develop a proof-of-concept prototype of an open platform for adaptive services discovery, creation, composition, and enactment. To proof the concepts identified and developed for and in ASG the technical realization as well as the business relevance is verified. The concepts for service discovery, creation, composition and enactment are technically realized in the ASG platform. For business relevant scenarios like the Attraction Booking Scenario[1] an application using the ASG platform and services provided by the ASG platform for the application are developed.

This document presents the processes for engineering services that can be developed on an ASG platform and applications using the ASG platform. The processes are tailored to different project contexts. To this end, we first describe a general process for the two engineering processes, application engineering and service engineering, and develop a number of scenarios that cover the different situations that can arise when an ASG platform is used. The scenarios provide customizations to the general process, so that the process fits optimally to a given project. The scenarios vary in different characteristics:

- Does the present domain ontology cover the semantic information about the service or the application, respectively
- Are new services developed to support an application or are the required services present already
- Are existing services or applications reused

We describe the different scenarios that result from combining these characteristics and present the respective process customizations.

The results presented in this document are on one hand the identified scenarios that capture the situation in which the end service providers and service providers can use ASG, and on the other hand the customized processes that end service providers and service providers can enact to perform application and service engineering in ASG-based software development projects.
This deliverable contributes directly to the ASG key feature "Seamless integration of heterogeneous external services" by providing methodological support for engineering applications that enable the identification and integration of external services.

This document can provide valuable input mainly to work component 7, in which the end service providers and service providers of the ASG project are located.

Table of Contents

1 Introduction

The goal of Adaptive Services Grid (ASG) is to develop a proof-of-concept prototype of an open platform for adaptive services discovery, creation, composition, and enactment. To proof the concepts identified and developed for and in ASG the technical realization as well as the business relevance is verified. The concepts for service discovery, creation, composition and enactment are technically realized in the ASG platform. For business relevant scenarios like the Attraction Booking Scenario[1] an application using the ASG platform and services provided by the ASG platform for the application are developed.

For the proof-of-concept prototype development we have specified an engineering process subdivided in three sub-processes as presented in Figure 1.

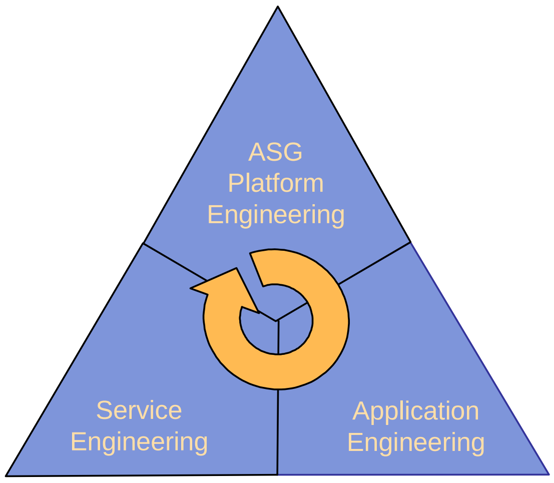

Figure 1: ASG Development Process

Naturally the prototype development and thus the process definition are highly driven by the evolution of the ASG project. Our goal regarding the process definition is to support the current development within the ASG project, but also the tailoring of ASG to a new application domain as described in [2]. The model presented in Figure 2 is the basis for the application of ASG to another domain: the end service provider owns an instance of the ASG platform and offers applications in the new domain running on the ASG platform. The services registered in the ASG platform and used by the application are provided by the service provider. The new application is used by the end service customer.

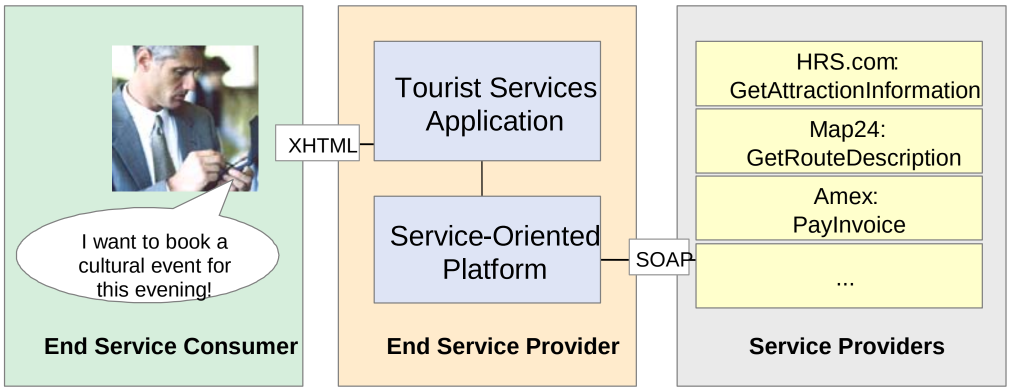
I want to book a cultural event for this evening!
End Service Consumer
XHTML
Tourist Services Application
Service-Oriented Platform
End Service Provider
SOAP
HRS.com: GetAttractionInformation
Map24: GetRouteDescription
Amex: PayInvoice
...
Service Providers

Figure 2: ASG Interaction/Communication Model

How the end service provider has to develop an application executable on the ASG platform and how the service provider has to develop a service to be registered in the ASG platform is specified in the application and service engineering processes presented in this report.

To get an impression on the challenges for application and service engineering processes the ASG platform, applications and services are characterized in the following sections.

1.1 ASG Platform

The service lifecycle for ASG as presented in Figure 3 illustrates and explains the ASG key service concepts and features.

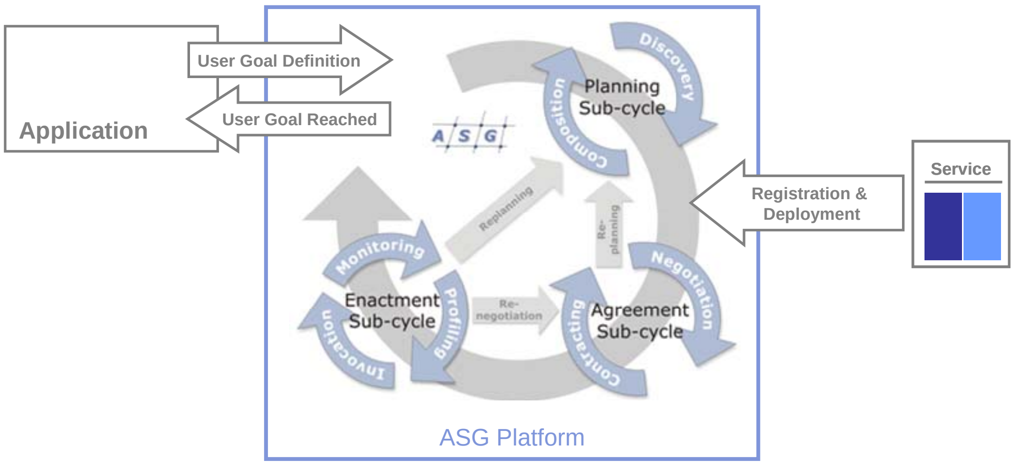

Figure 3: ASG Lifecycle

The main feature of the ASG platform is to provide services corresponding to the specified user goal. During planning the composition subsystem of the ASG platform looks for single (atomic) services or a composition of services (composed services) achieving the user goal. It uses the discovery subsystem to search the current service landscape for an appropriate service. In order to discover, compose and subsequently to compare available services or their composition with the user goal automatically at runtime the services are described semantically. In contrast to that is the, current state-of-the-art, where services have to be composed manually at design time due to the lack of semantic information in their service specifications.

After the planning has found or composed an appropriate service, an agreement on quality of service is made with provider of the service identified for achieving the user goal. The negotiation considers the user preferences regarding quality of service as well as the offers of providers. These service level agreement (SLA) is documented in a contract.

The enactment of the agreed service is managed and controlled by a workflow engine. It invokes each identified service (according to the composition), monitors if the contract is fulfilled by each provider, and subsequently specifies or updates the profile of the provider according to the agreement and monitoring results. The output of the executed service is returned to the requester.

In each sub-cycle of the ASG platform lifecycle the possibility for regression exists. If for example no agreement is accomplished, composition is triggered again (re-planning).

Usually, as illustrated in Figure 3, an application is used to build goals based on user input and thus requests the ASG platform for an appropriate service by specifying the (user) goal. The application, its features and its usage of the ASG platform is described in section 1.3.

Further, any kind of functionality deployed as service (see chapter 4) can be registered at the ASG platform. The registration requires a semantic and syntactical specification of the service according to the ASG ontology and domain ontology (see also *Service Engineering - Register Service* in section 3.2).

The process to develop the ASG platform with its features (e.g. dynamic composition, dynamic negotiation, semantic awareness …) supported by the lifecycle and its different subsystems is described in [3].

1.2 ASG Service

The ASG platform discovers, composes, and negotiates appropriate services with semantic-awareness and subsequently executes them. The semantic-awareness is based on the semantic specification as well as on the underlying domain ontology. To execute services they are invoked based on their syntactic specification. Thus, each service is specified semantically and syntactically (see Figure 4). The specification comprises the attributes as defined in the ASG Ontology and their values defined according to concepts in the domain ontology.

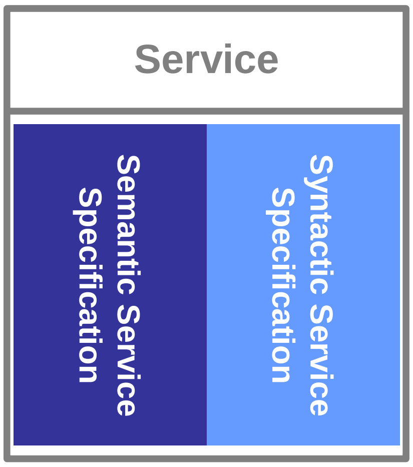
Service
Semantic Service Specification
Syntactic Service Specification

Figure 4: ASG Service

The ASG Ontology comprises the attributes required to specify a service semantically and syntactically and their relationships. For the semantic specification, attributes like precondition and postcondition are required, and the syntactic specification contains attributes like input, output and operation name.

The domain ontology is formal and consensual specification of conceptualizations that provide a shared and common understanding of services registered at the ASG platform.

The processes to engineer or reengineer services are described in chapter 3 and 4.

1.3 ASG Application

Applications using the ASG platform are service-oriented. The functionality of the application, the Application Logic, is described by a flow of services as shown in Figure 5. The functionality required by the services is offered by the ASG platform, by another service registry or by an internal component repository. According to the service functionality required in the application flow, the goal is defined in the Service Request layer and transmitted to the ASG platform. The input required for performing the services are received by the application via the User Interface. The results of the service execution are propagated to the user also via this layer. In chapter 3 and 4 the development of an application are described in more detail.

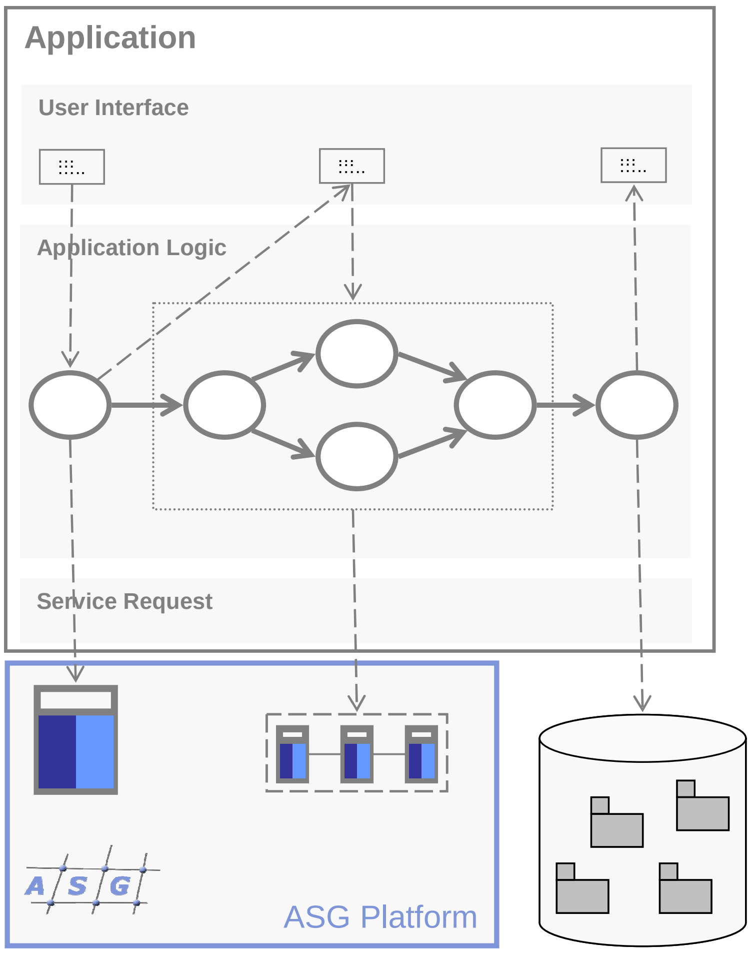

Figure 5: ASG Application

1.4 ASG Process Management

In this section we describe the approach used to develop the platform, applicatin and service engineering processes. Figure X shows the five main phases of the approach: Initial Process Drafting, First Process Refinement, Second Process Refinement, Process Stabilization, and Post-Mortem Analysis.

In general, the main idea behind the approach followed is to start with commonly accepted process knowledge, to refine it with information gathered from the development cycles, and to improve therewith the process according to the real project needs. For more details see [3].

Initial Process Drafting. During this phase, a first process draft is sketched on the basis of recognized international standards such as ISO/IEC12207 "Information Technology - Software Life Cycle Processes" [4], and 15504 "Information Technology – Software Process Assessment" [5] and discussed with the responsibles of the development to ensure its applicability. The main goal of this phase is to make the various process purposes explicit and provide a first set of process-related terms to be used by the heterogeneous teams involved in the development of the ASG platform, applications and services.

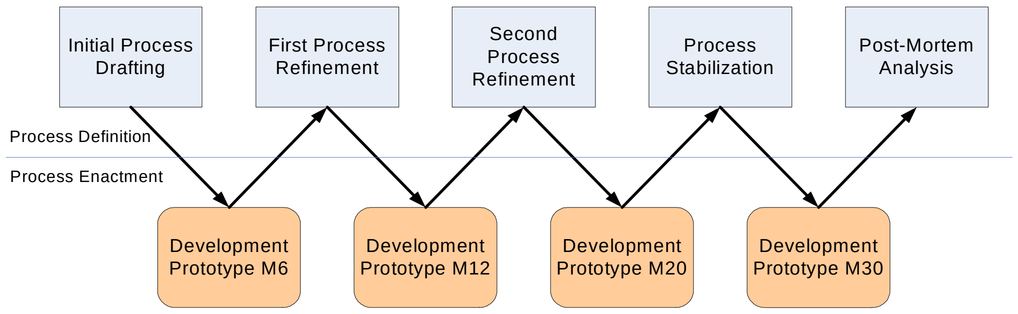

Figure 6: Approach to Process Management

First Process Refinement. In this second phase, documents produced during the first development cycle are analyzed and the process description is refined accordingly. Process refinements at this stage include more detailed guidance for specific activities and tools. A first formalization of process roles is achieved and responsibilities are tentatively assigned. The process infrastructure is also refined, since first document templates are defined and the project portal is (re)structured to reflect, among other things, the process information gathered. The purpose of the second phase is to improve process awareness and let the heterogeneous development teams come to a common process understanding.

Second Process Refinement. The purpose of the second process refinement is to let all parties involved in development achieve a common understanding of the development process. For this purpose, a detailed process description is provided, which reflects the development activities as performed by the involved roles, and which is agreed on by all process performers. During this phase, project members involved in the development of the ASG platform, application and services are interviewed. They answer questions about their experience and the role they play, the experience level and the organisation of their software team, the process followed by the team. Also, they provide feedback on the process as refined during the second phase and on the process infrastructure implemented.

Process Stabilization. During this phase, process changes are controlled through a change management procedure: changes must be requested, motivated, analysed, and collectively accepted before they can be implemented. The purpose of this phase is to keep the process stable to enable unbiased process analysis. Nevertheless, valuable process changes should be considered and eventually implemented whenever needed.

Post-Mortem Analysis. At the end of the project, a post-mortem analysis will be performed by interviewing process performers. The purpose of this closing phase is to extract lessons learned during development and to package the main project results, also in terms of considerations regarding the process applied.

1.5 Structure of this Report

This report presents the processes required to engineer services that can be developed on an ASG platform and applications using the ASG platform.

Chapter 2 summarizes the current state of the practice in application and service engineering. The activities, artefacts, roles and tools needed to engineer a new application and services used in the application are described in chapter 3. Chapter 4 lists and describes possible scenarios using the ASG platform as end service provider and service provider. For each scenario the application as well as the service engineering process is instantiated correspondingly.

2 Related Work

This section introduces processes, methods, and techniques available for application and service engineering. The section sketches the current state of the practice in the field of processes aimed at providing solutions based on service-oriented architecture with particular attention given to the initiatives currently exploring semantic interoperability aspects. The content of this section is a summary of the survey performed at M6 and documented in deliverable *D6.III-1 "Adaptable Process Engineering Survey"*.

2.1 Application Engineering

The task of developing an application for the ASG platform is characterized by factors such as: high volatility of requirements, lack of experience, and time to market. Software development organizations that target developing in domains of similar characteristics must have the possibility to change their strategy in a given circumstance, in a given point of time during development. Strict, inflexible process models like the waterfall model are not suitable for such contexts. Organizations must react to the context in the most appropriate manner, and that is only possible through flexible processes [6].

Suitable life cycle processes for developing applications for the ASG platform are:

The throwaway prototype model, and the incremental development model [7], which were found suitable for domains of similar ASG characteristics such as Internet and mobile [8], [9], [10]. Through these models, essential operational functions are provided initially, and then more capable versions of the system. Increments are usually defined as an agreement between the customer and the development organization. This allows development organizations to get feedback from the final customer during the development of the increments until the final version of the solution is delivered. Additionally, monitoring and controlling the project plan can be done more precisely, and the quality of increments can be assured with the established verification and validation activities. What is the best increment to be delivered? What is a realistic time interval for each increment? How to select a consistent set of requirements for the increment? These are questions, which have been already addressed in the area of requirements engineering and applied in areas like Internet or Mobile [11], [12].

Regarding applications to be integrated in the ASG platform, a software development organization can have more than one role, for example as application provider, service provider, and integrator of services. This characteristic demands vision and discipline for releasing the product, and impacts the procedure to determine the best suitable or most profitable increment. In the context of the ASG project prototype development has been used to developed the applications and realize the business scenarios.

The Extreme Programming approach does also reflect an incremental model and has been proposed by [13] and [14] as suitable for Web based projects where time to market plays an important role. Extreme programming focuses on producing source code and test drivers, avoiding documentation, and handling the volatility of requirements through small releases. Development cycles are short and based on requirements that will really generate business value for the customer. A risk of following agile approaches is that they rely on tacit knowledge of developers [15]. In the context of the ASG this can become an issue especially because developers are still learning due to the immaturity of the Semantic Web Services domain. Additionally, issues like scalability and performance have to be carefully designed.

The spiral model [16], assumes risks as the driver force of software projects. This model proposes ongoing refinement of the system specification into source code components. Refinements are made through cycles, and each cycle is risk assessed. A risk assessment determines if a project continues or is cancelled. The nature of the spiral model seems reasonable to apply in a convulsionate domain like semantic web services, but the real cost of identifying, analyzing and maintaining risks is high, which is not so suitable for small and medium companies, and the ASG project infrastructure.

According to [17], it is advisable for any organization intending to engineer an application for a platform of similar characteristics as the ASG to make explicit decisions at the management level before the start of the project concerning the following issues:

- Which types of benefits would the organization like to achieve first?
- What scope does the organization want for the platform-based application in the near and medium term?
- Which technical aspects or elements of a platform-based application should be exploited first?

These questions have been and continue to be systematically answered in the context of the ASG project before developing the prototypes. Benefits have been realized by business scenarios, the scope through the requirements of the ASG platform, and the technical aspects to be exploited first through the prioritization of the requirements. [18] proposes one strategy for establishing the foundations of a grid service application. The strategy comprises the following set of steps:

1. Identify similar resources from a business perspective (i.e., potential Web or telematic services),
2. Handle services in a semantic oriented way,
3. Virtualize such services. In this context, virtualization means making information available whenever, wherever it is needed [17], [19].

ASG has followed the first two steps of such a strategy. The third step has not been followed since grid computing has been left out of the scope of the project. In the following section, we present approaches found in literature for engineering Web and telecommunication services that cover the first steps of the mentioned strategy. We also reference approaches found for providing semantic capabilities to services, which corresponds to the second step of the strategy.

2.2 Service Engineering

Web and Telecommunication Services - approaches that can be of help in identifying similar resources from a business perspective can be found in the Web service-oriented engineering domain as well as in related domains such as telecommunications [20], [21] ,[22],[23].

These domains do not show substantial differences in their processes when transforming a business idea into a service model. They suggest capturing the business idea through scenarios, then creating a conceptual service model that reflects service concepts involved in the mentioned scenarios, followed by a refinement of the service flow by defining operations, relationships to external services, and states of the service, and finally, orchestrating the service by defining rules and interaction models. These steps have been followed for the development of ASG prototypes and captured in the ASG development process.

Traditional requirements analysis techniques can be performed through interviews or group meetings with stakeholders in order to discover candidate services. Another possibility is to follow the Component Business Modelling (CBM) technique [24]. CBM is a technique that could help in deriving services in a top-down manner. It provides a framework for viewing the business as a network of discrete services, turning the services into unique building blocks. A comprehensive survey on approaches for eliciting candidate services and specifying them as requirements is provided in *D6.I-1 Requirements Specification Survey*. The survey is used as a basis for developing the requirements engineering method for ASG.

In ASG, services are identified and discovered in an application-specific way. This means that, for an envisioned application that solves a given problem, during application engineering, services are identified that contribute to the application. The service identification is driven by the application flow that captures the business logic, as well as the interaction of a user with the application. Blocks of interaction are refined until a proper level of granularity is reached and thus services have been identified. The identified services are then discovered to check whether there are actual service implementations available. If this is not the case, service engineering is enacted to develop an appropriate service. In the EU IST project SeCSE, two different approaches for service discovery have been developed. The first approach integrates the service discovery in the requirements specification [25]. The idea of this approach is to use relevance feedback to use existing services to explore and validate the requirements for service-centric systems. A tool is developed that supports the use of UML use cases and textual requirements for requirements specification on one hand and for generating service requests that are used for service discovery. The tool uses a word-net based glossary to generate service requests from natural language requirements.

The second approach proposes architecture-driven service discovery [26]. In this approach design models are used to identify services. The functional and quality requirements for the services are thereby taken from the design models. The used design models are UML structural and behavioural model. From these service queries are derived that are used to discover available services. As in the first approaches described before, the discovered services are fed back to evolve the design.

Both approaches developed in the SeCSE project have in common that they discover services at development time. In the ASG application engineering services are discovered as well at design time to support the refinement of interaction blocks. ASG does, however, also support the dynamic and automatic service discovery based on semantic service and service request descriptions.

Business process models can be described after the candidate services have been identified. They shall be described as a sequence of operations/services performed with a specific business goal in mind. Once the business process model has been identified, techniques from enterprise architecture frameworks and object-oriented analysis and design can be used for implementing and deploying the service(s) [20]. The actual tendency of major software vendors (e.g., Websphere Integration Server Foundation, Business Works, Oracle BPEL Process Manager) is to provide support for the static and dynamic design of such business processes as well as for their implementation. Software vendors enable the definition of state models and choreography of business processes. They also integrate Web services with process engines [27], [28] on top of their Web application servers, e.g., the IBM WebSphere Application Server - Express V5.0.2. The ASG project is using the advantages offered by such tools to represent executable flows of models, i.e., the choreography, and object-oriented analysis and design for implementing and deploying the services.

Semantic Services - Concerning the engineering of solutions aimed at handling services in a semantic-oriented way, four different approaches are currently evolving: IRS [29], OWL-S [30], WSMF [31] or rather WSMO [32], and METEOR-S [33]. Although they address similar objectives, they also turn out to be different in terms of reasoning support, mainly due to different underlying logic and ontology frameworks. The approaches show complementary strengths and there is evidence of convergence among the approaches. However, none of the proposed frameworks can be seen as a reference capable of supporting standard processes for the semantic annotation of services and the engineering of intelligent machines able to autonomously apply and combine the annotated services for coming up with a solution to a given problem.

WSMO and its language WMSL have been adopted by the ASG project for enriching services with semantic descriptions. They have been used and continue to be used for developing the prototypes. The ASG project profits from the existing concepts and constructs developed in WSMO and WSML. On the other hand, ASG generates new requirements to be implemented by the leaders of the WSMO initiative.

3 Application and Service Engineering Process

In this chapter the application and service engineering processes with all their possible activities to be performed, artefacts used and produced in these activities, the involved roles and the tools supporting the processes are presented. How the end service provider has to develop an application executable on the ASG platform and how the service provider has to develop a service to be registered in the ASG platform is specified in these processes.

As described in section 1.1 the ASG platform provides appropriate services by discover and compose them based on semantics. On one hand the services are described semantically. On the other hand the ASG platform comprises a domain ontology specifying the concepts of applications using the ASG platform and of services registered at the ASG platform, the service landscape.

The following parts of the ASG platform could be updated or developed for new applications and services:

- **Domain ontology:** the ontology specifying the domain of applications and services available in the ASG platform will be updated/developed in case of required domain changes regarding new applications or services.
- **Service Registry:** the registry managing services available currently in the ASG platform can be extended or reduced by adding or removing services.

Figure 7 presents the product flow of service and application engineering process. The product flow describes all activities and the artefacts which are used as input for the activity and which are produced as output of the activity (see legend in Figure 7). The notation is based on spearmint [34].

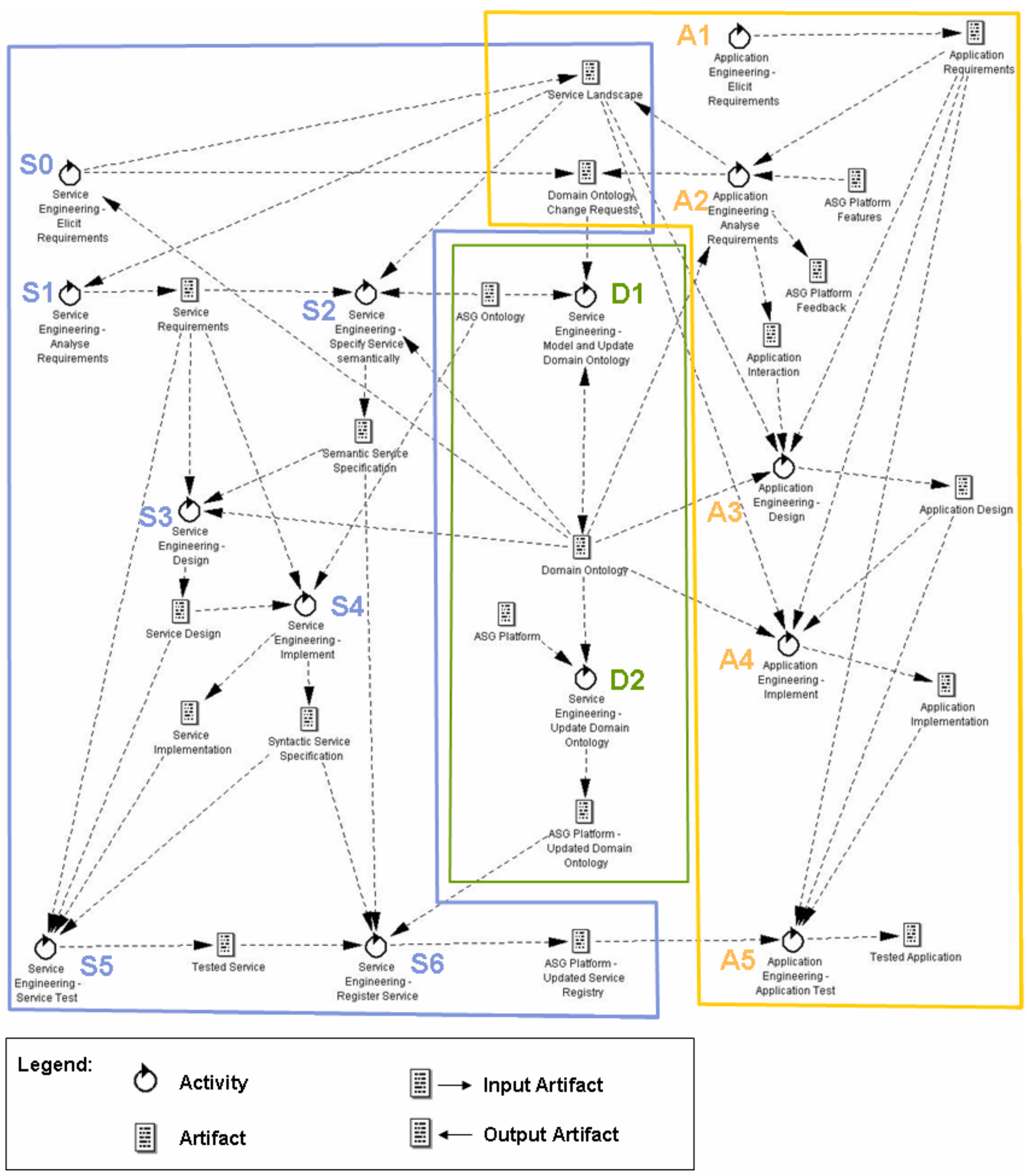
A1
Application Engineering - Elicit Requirements
Application Requirements
Service Landscape
S0
Service Engineering - Elicit Requirements
Domain Ontology Change Requests
A2
Application Engineering - Analyse Requirements
ASG Platform Features
ASG Platform Feedback
Application Interaction
S1
Service Engineering - Analyse Requirements
Service Requirements
S2
Service Engineering - Specify Service semantically
ASG Ontology
D1
Service Engineering - Model and Update Domain Ontology
Semantic Service Specification
S3
Service Engineering - Design
Domain Ontology
A3
Application Engineering - Design
Application Design
Service Design
S4
Service Engineering - Implement
ASG Platform
A4
Application Engineering - Implement
Application Implementation
D2
Service Engineering - Update Domain Ontology
Service Implementation
Syntactic Service Specification
ASG Platform - Updated Domain Ontology
S5
Service Engineering - Service Test
Tested Service
S6
Service Engineering - Register Service
ASG Platform - Updated Service Registry
A5
Application Engineering - Application Test
Tested Application
Legend:
Activity
Artifact
Input Artifact
Output Artifact

Figure 7: Product Flow – Application and Service Engineering

Performing service and application engineering processes as presented in Figure 7 services, application, ASG platform and the domain ontology are developed or rather updated. This situation is illustrated in Figure 8. Thus, new application and its required services are developed. The domain ontology is updated according to the new service landscape, in other words according to the concepts required to describe the semantic of the application or rather services. The new ontology is integrated in the ASG platform and subsequently the new services are registered. In addition, feedback regarding future features or other recommendations could be collected for the ASG platform.

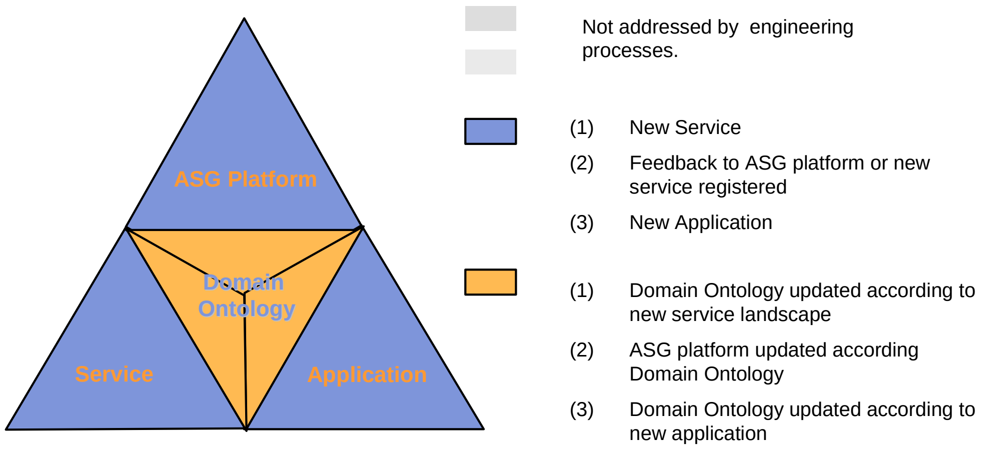
ASG Platform
Domain Ontology
Service
Application
Not addressed by engineering processes.
(1) New Service
(2) Feedback to ASG platform or new service registered
(3) New Application
(1) Domain Ontology updated according to new service landscape
(2) ASG platform updated according Domain Ontology
(3) Domain Ontology updated according to new application

Figure 8: ASG Elements

3.1 Activities – Application Engineering

In this section each activity that have to be performed to develop an application (see Figure 7) using the ASG platform is described by its purpose, used and produced artefacts, involved roles and tools supporting the activity.

The first activity in application engineering is *Elicit Requirements* (see Figure 7 A1). The purpose of this activity is to identify the high-level requirements to be fulfilled by the intended application. Therefore, the different stakeholders interested in this application and their goals regarding this application are identified. The result is the artefact *Application Requirements*. *Application Requirements* is a text document containing the goals intended with the application by any stakeholder. For more details see [35]. To identify the right and feasible requirements the *Customer*, *Application Engineer*, and an *ASG Platform Expert* should participate. *ASG Platform Expert* has the knowledge on the functionality and features provided by the ASG platform. The feasibility of the intended application has to be estimated by the expert. *Application Engineer* is familiar with the application engineering process. He/She knows how to perform the different steps and to use as well as to produce the required artefacts. The role *Cus-*

tomer is used in the description of service and application engineering process more generic. It is assigned to the person that has interest in the development. Thus, it could be assigned to the end service provider, end service customer and service provider or to any other stakeholder. The customer involved in this activity could be the end service provider as well as the end service customer or both. To document the requirements any kind of text editor, for example MS Word, Wiki, can be used.

The next activity (see Figure 7 A2), *Analyze Requirements* refines the high-level requirements. First, the flow of activities within the application and the interactions are identified and analyzed if the flow and interaction can be supported by the functionality of the ASG platform. Concurrently, services are investigated providing the functionality of activities identified in the workflow. These services are either already registered at the ASG platform or have to be developed (see section 3.2). Further, possible changes required in the domain ontology are optionally found out considering the domain of the intended application [35]. Inputs for the refinement of the requirements are the *Application Requirements*, the *Domain Ontology* and *ASG Platform Features*. *Domain Ontology* comprises all application domains. Thus, it could already include an ontology specification for other applications. *ASG Platform Features* contains the functionality and features provided by the ASG platform like a user manual. Outputs are the *Application Interaction* specification, *Domain Ontology Change Requests*, *Service Landscape* and optionally *ASG Platform Feedback*. *ASG Platform Feedback* is a document comprising requirements to the next release of the ASG platform. *Application Interaction* describes activities, their sequence interactions used to realize the identified application requirements. *Service Landscape* contains the list of services used to support partially the application flow. Services already registered in the ASG platform as well as services that have to be developed are listed. They are mainly described by their input, output, precondition, postcondition and their functionality. *Domain Ontology Change Requests* lists the changes needed for the domain scoped during *Application Engineering – Analyse Requirements*. The templates used to specify *Application Interaction*, *Service Landscape* and *Domain Ontology Change Requests* are presented in [35]. The *Application Engineer*, *Customer*, *Service Engineer* knowing the engineering process and already existing services, *Domain Engineer* knowing the current domain ontology, and the *ASG Platform Expert* analyzing the feasibility to support the application by the ASG platform should participate in this activity. To document the application flow a workflow specification editor and to specify the other artefacts a text editor can be used.

After the requirements analysis the realization of the application is designed (see Figure 7 A3). As described in section 1.3 the application comprises a flow of activities supported by services already registered in the ASG platform, by services to be developed, or any internal components. Further, the application handles the activity input data given by the end service customer and output data represented to the end service customer. Considering these application features, a generic layered design for the application can be specified as presented in Figure 9.

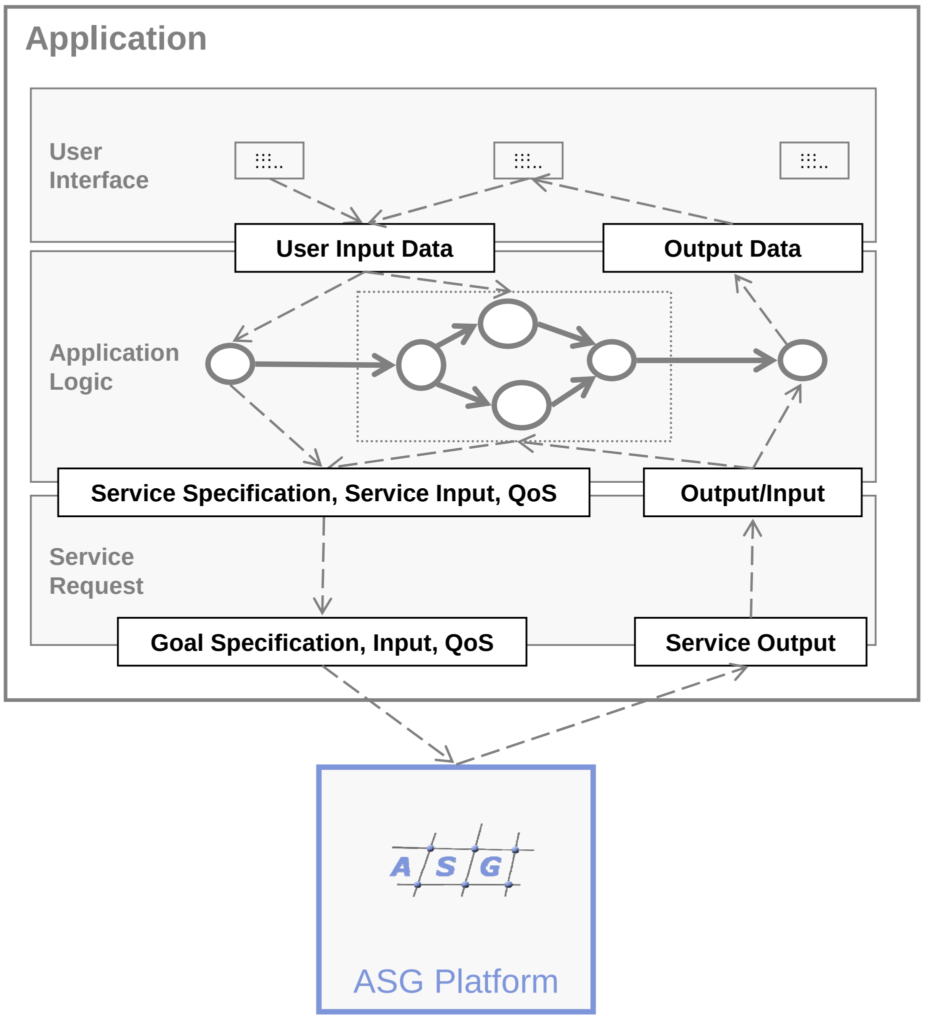

Figure 9: Application Design

The user interface layer provides the interfaces to specify the input data as well as to present the output to the user. TORE is a method that allows developing the user interface based on the application flow description [36]. The application logic layer has the functionality to execute the application flow. It performs the activities one after another by calling the ASG platform providing an appropriate service or internal components. The functionality of the application logic layer can be realized by a workflow engine, for example. Thus, the workflow/application flow is specified using for example BPMN or UML diagrams describing the flow. The service request layer specifies on the one hand the goal as input for ASG platform request and on the other hand the functional call for the internal component. The functionality of the internal components is designed using regular software engineering approaches. Inputs for the application design activity are the *Application Interaction*, *Domain Ontology*, *Application Requirements* and *Service Landscape*. The *Application Engineer*, *Domain Engineer* as well as the *ASG Platform Expert* should be involved in the design. For the documentation of the application design any design or UML editor can be used.

After the design activity an executable application is established (see Figure 7 A4). The purpose of the activity *Application Engineering – Implement* is to implement the functionality of the design layers using the selected technology as programming language, workflow engine, and of the internal components and subsequently, to compile the resulting source code. The *Application Design*, *Service Landscape*, *Domain Ontology*, *and Application Requirements* are used as input for the implementation. Output is the *Application Implementation*. *Application Implementation* is executable source code implementing the application functionality and behaviour as described in the application design. Responsible for this activity is the *Application Engineer*. The implementation can be supported by any integrated development environment (IDE), for example Eclipse and Maven.

Concluding, the functionality of the resulting application implementation is verified against the specified *Application Requirements* (see Figure 7 A5). For more details on the *Application Engineering – Test* see [37]. Input for this activity is the *Application Implementation*, *Application Requirements, Application Design* as well as *ASG Platform with Updated Service Registry*. *ASG Platform - Updated Service Registry* is the ASG platform with new uploaded semantic and syntactic service specifications. The result is an executable and *Tested Application*. *Tested Application* is the application tested against the application requirements. The *Application Engineer* is responsible for the test.

3.2 Activities – Service Engineering

In this section each activity that have to be performed to develop services (see Figure 7) that can be registered in the ASG platform is described by its purpose, used and produced artefacts, involved roles and tools supporting the activity.

Considering service engineering as a stand-alone project independent from a specific application (see section 4.1) the first activity in service engineering is Elicit Requirements (see Figure 7 S0). The purpose of the activity is to identify the service to be developed. The result is a *Service Landscape* with the service described by its name, input and output parameters as well as the specified provider. If changes are required in the domain ontology according to the identified service landscape, they are documented in the artefact *Domain Ontology*

Change Requests. The *Service Engineer*, *Domain Engineer* as well as the *Customer* are involved in this activity. *Service Engineer* is familiar with the different steps to develop a service and with the usage and documentation of specified artefacts. Further, he/she has the knowledge to specify a service syntactically and semantically. The results are documented with usual text editors. *Domain Engineer* is an expert in modelling and specifying ontology. He/She is familiar with the ASG Ontology and the ontology language(s) understandable by the ASG platform like WSML or Flora[1].

However considering the scenario that services for a specific application and corresponding domain ontology have to be developed (see section 4.2.4) service engineering starts with the activity *Analyze Requirements* (see Figure 7 S1). The purpose of the activity is to identify the external visible behaviour of the service. Thus, the functionality callable from extern, the interaction among service and user/application, are analyzed in more detail, e.g. the data types used for input and output paramaters are specified. Input for the activity is the Service Landscape. The result is *Service Reuqirements* document. The *Service Engineering* as well as the *Customer* have to participate.

Semantic-awareness is one feature of the ASG platform. The purpose of the activity *Specify Service Semantically* (see Figure 7 S2) is to annotate the service semantically regarding their affiliations to the domain ontology. First, the functional and non-functional properties identified in the Service Requirements are specified using the established service specification language, for example WSML or Flora. The functional and non-functional properties that have to be specified for ASG platform are defined in the *ASG Ontology*. For the service specification the *Service Landscape, Service Requirements* and the *Domain Ontology* are considered. The output of this activity is the *Semantic Service Specification* for the developed service. *Semantic Service Specification* is a document specifying the semantic of the service. The elements like input, output, precondition, and postcondition as specified in the ASG Ontology are described according to the domain ontology. The specification task is performed by the *Domain Engineer* and *Service Engineer*. They could use specific tools for the semantic annotation of services as WSMO studio [39].

In the next step the service is designed (see Figure 7 S3). The purpose is to identify and define the internal behaviour of the service. The internal components required to provide the external visible functionality, the data exchanged within the service, as well as the interaction among the internal components of the service. For the documentation corresponding UML models are used:

[1] In the current ASG platform Flora is used. However, the goal of ASG is to use WSML as soon as an appropriate reasoner available, which is an ASG project result.

- Internal structural service model: UML class diagram
- Interaction among internal service components: UML sequence diagram
- Exchanged Data among internal service components: UML activity diagram

The design activity uses the specified *Service Requirements*, *Semantic Service Specification* and *Domain Ontology* as input. The output is managed as *Service Design* artefact. *Service Design* is a document describing internal structure and behaviour of the service. Any UML modelling tool, for example MagicDraw [40], supports the documentation of the service design.

After the design step an executable service is established (see Figure 7 S3). According to the design the service is implemented and compiled using the underlying technologies, for example Java. As the important feature and advantage of services is their standardized interface a syntactic service description using for example WSDL is generated for the developed service. The specification required in the ASG platform is described in the ASG Ontology. Input for the activity as mentioned above are the *Service Requirements*, *Service Design* and the *ASG Ontology*, and the result is a *Service Implementation* and *Syntactic Service Specification*. *Service Implementation* is executable source code implementing the service functionality and behaviour as described in the service design. *Syntactic Service Specification* is a document specifying the syntax of the service. The elements like port, name, and operation as specified in the ASG Ontology are described. Responsible for that activity is the *Service Engineer*. Any kind of integrated development environment (IDE), for example Eclipse, supports the service implementation.

Now, a test validating the service implementation is performed (see Figure 7 S5). The purpose is to validate the service implementation against the service requirements, service design, syntactical service specification, and semantic service specification. For more details on the *Service Engineering – Test* see [37]. Input for this activity is the *Service Implementation*, *Service Specifications*, *Service Requirements, Service Design* as well as *ASG Platform with Updated Service Registry*. The result is a runable and *Tested Service*. *Tested Service* is the service tested against the service requirements, syntactic and semantic service specification and callable by the ASG platform. The *Service Engineer* is responsible for the test.

As described in the service engineering activities (S0 – S5) the domain ontology is often used as input. However, the results of the different engineering activities have also impact on the domain ontology. Thus, the activities S0 to S5 and D1 to D2 are sub engineering processes that will be performed in parallel and with data exchange many a time.

The purpose of the activity *Model and Update Domain Ontology* (see Figure 7 D1) is to specify new domain ontology or to update the existing one by extension (or by integrating mediation). The first step is to model the ontology with its different functional concepts and their relationships. Then, this model is translated into the ontology language used in the ASG platform, for example WSML or Flora. The elements of the domain ontology required in the ASG platform are specified in the ASG Ontology. Input for this activity is the *Service Landscape* describing the service requirements and optional the *Existing Domain Ontology*, *Domain Ontology Change Requests* as well as the *ASG Ontology*. *ASG Ontology* consists of two models one specified the elements within domain ontology and their relationships, and the other one describes the elements required for the syntactic and semantic service specification. The models should be described for example in the user manual of the ASG platform. The result is the new *Domain Ontology*. *Domain Ontology* describes the concepts used in this domain and their relationships using the established ontology language, for example WSML or Flora. The specification or update is task of the *Domain Engineer*. To model the concepts of the ontology the tool CMAP [38] is used. The identified concepts could be subsequently specified in WSML using WSMO studio [39].

Considering the parallel work of the subprocesses S0 to S5 and D1 to D2 the activity *Update Domain Ontology* takes place after *Service Test* is finished (see Figure 7 D2). The purpose of the activitiy is to replace the existing domain ontology with the new domain ontology. An interface provided by the ASG platform is used to insert the domain ontology. Afterwards, the ASG platform with updated domain ontology is available. *ASG Platform – Updated Domain Ontology* is the ASG platform with new defined domain ontology. Responsible for the update is the *Domain Engineer*. As mentioned, the activity is supported by the ASG platform itself.

Concluding the service engineering process, the new developed service can be registered at the ASG platform (see Figure 7 S6). Again, an interface provided by the ASG platform is used to insert the syntactic and semantic service specification in the service registry. Thus, input for the activity *Register Service* is the *Tested Service* itself, *Semantic Service Specification*, *Syntactic Service Specification* and the *ASG Platform – Updated Domain Ontology*. The result is the *ASG Platform* with *Updated Service Registry*. *ASG Platform - Updated Service Registry* is the ASG platform with new uploaded semantic and syntactic service specifications. Responsible for the activity is the *Service Engineer*; the task is supported by the ASG platform itself.

4 Usage Scenarios for ASG Engineering Processes

In this chapter different scenarios to tailor Adaptive Service Grid to a new application domain, in other words, to use the ASG platform are described. Each identified scenarios requires a specific instance of the application and service engineering process. In following sections each scenario is specified by a description of the situation, approach to tailor ASG for this scenario, and the required engineering process.

4.1 Service Engineering

4.1.1 Service Semantic covered in Existing Domain Ontology

Description of the Situation

A service provider wants to provide a service that can be registered at and offered by the ASG platform. This service first has to be developed and subsequently specified, so that the service could be used in ASG. In ASG the service have to be specified both semantically and syntactically. The domain ontology which exists already in the ASG platform comprises concepts to describe the functionality of the service semantically.

Approach

The service engineer of the service provider has to develop this new service and its semantic and syntactic service description. Subsequently, the new service with its specification is registered at the ASG platform. Figure 10 illustrates the tailoring of ASG according to this situation.

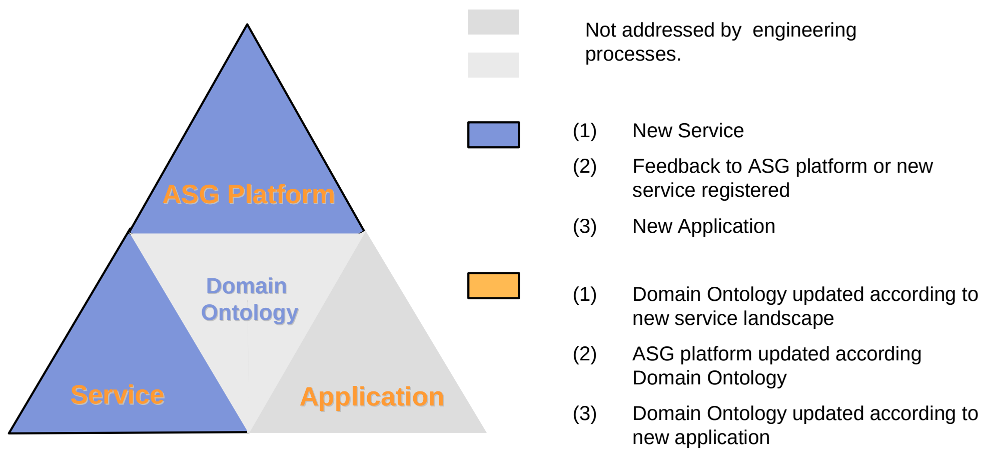
ASG Platform
Domain Ontology
Service
Application
Not addressed by engineering processes.
(1) New Service
(2) Feedback to ASG platform or new service registered
(3) New Application
(1) Domain Ontology updated according to new service landscape
(2) ASG platform updated according Domain Ontology
(3) Domain Ontology updated according to new application

Figure 10: Approach: Service Semantic covered in Existing Domain Ontology

The engineering process that has to be performed for the service development and the registration are described in Figure 11. The service engineering activities regarding the development of the domain ontology are not used in this situation.

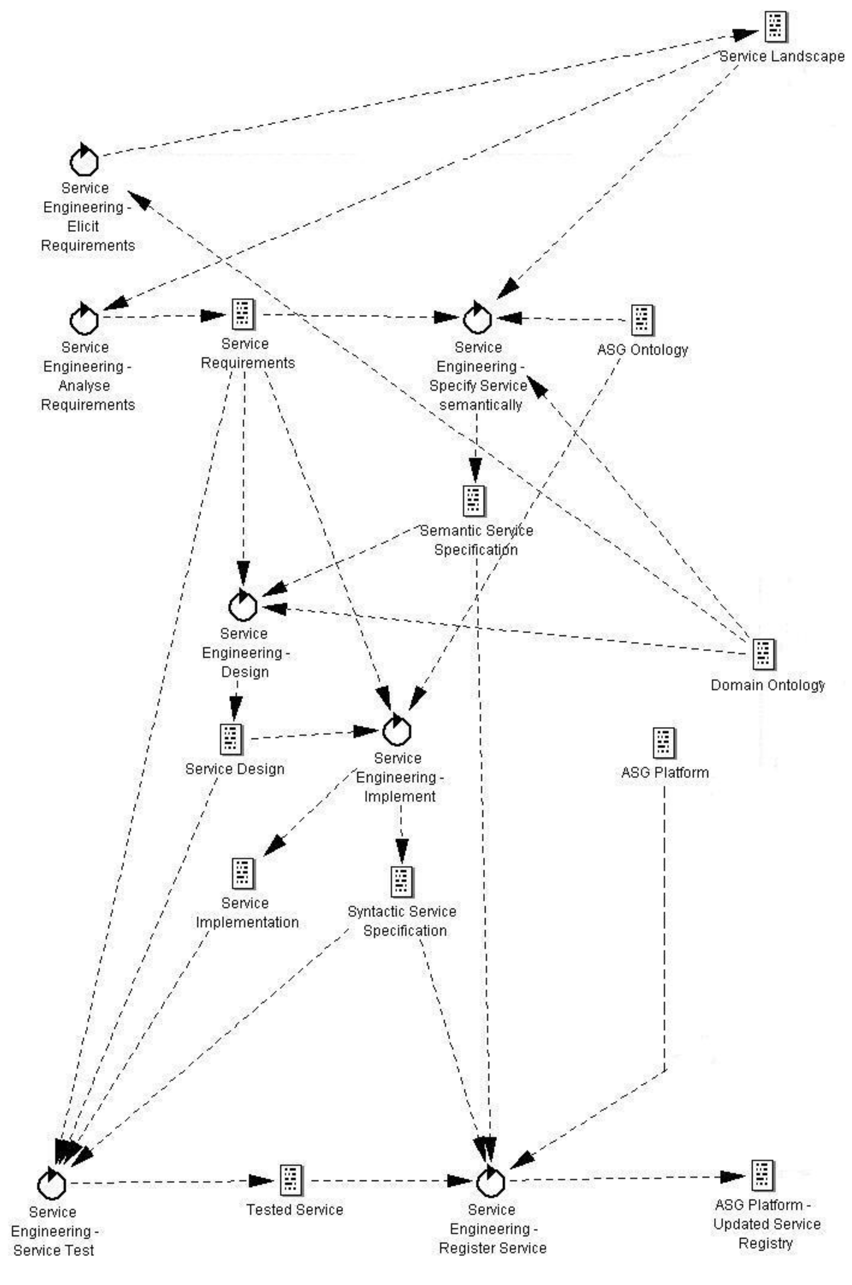
Service Landscape
Service Engineering - Elicit Requirements
Service Engineering - Analyse Requirements
Service Requirements
Service Engineering - Specify Service semantically
ASG Ontology
Semantic Service Specification
Service Engineering - Design
Domain Ontology
Service Design
Service Engineering - Implement
ASG Platform
Service Implementation
Syntactic Service Specification
Service Engineering - Service Test
Tested Service
Service Engineering - Register Service
ASG Platform - Updated Service Registry

Figure 11: Engineering Process: Service Semantic covered in Existing Domain Ontology

4.1.2 Service Semantic not covered in Existing Domain Ontology

Description of the Situation

A service provider wants to provide a service that can be registered at and offered by the ASG platform. This service first has to be developed and subsequently specified, so that the service could be used in ASG. In ASG the service have to be specified both semantically and syntactically. The domain ontology which exists already in the ASG platform does not comprises the concepts to describe the functionality of the service semantically, or no domain ontology exists yet in the ASG platform.

Approach

The service engineer of the service provider first has to develop the new service. Then, the domain ontology has to be updated or rather developed according to the new concepts required to describe the functionality of the service semantically wit the help of the domain engineer. Subsequently, the service and domain engineer specify the semantic and syntax of the service. Before registering the service and its specifications, the new domain ontology has to be integrated in the ASG platform. Concluding the service with its specification is registered at the ASG platform. Figure 12 illustrates the tailoring of ASG according to this situation.

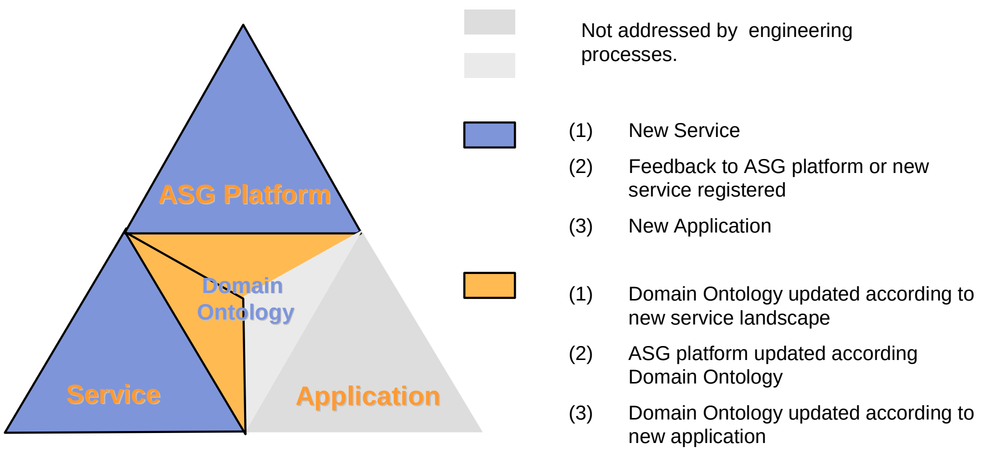
ASG Platform
Domain Ontology
Service
Application
Not addressed by engineering processes.
(1) New Service
(2) Feedback to ASG platform or new service registered
(3) New Application
(1) Domain Ontology updated according to new service landscape
(2) ASG platform updated according Domain Ontology
(3) Domain Ontology updated according to new application

Figure 12: Approach: Service Semantic not covered in Existing Domain Ontology

The engineering process that has to be performed for the service development, domain ontology update and the service registration are described in Figure 13. All service engineering activities have to be performed.

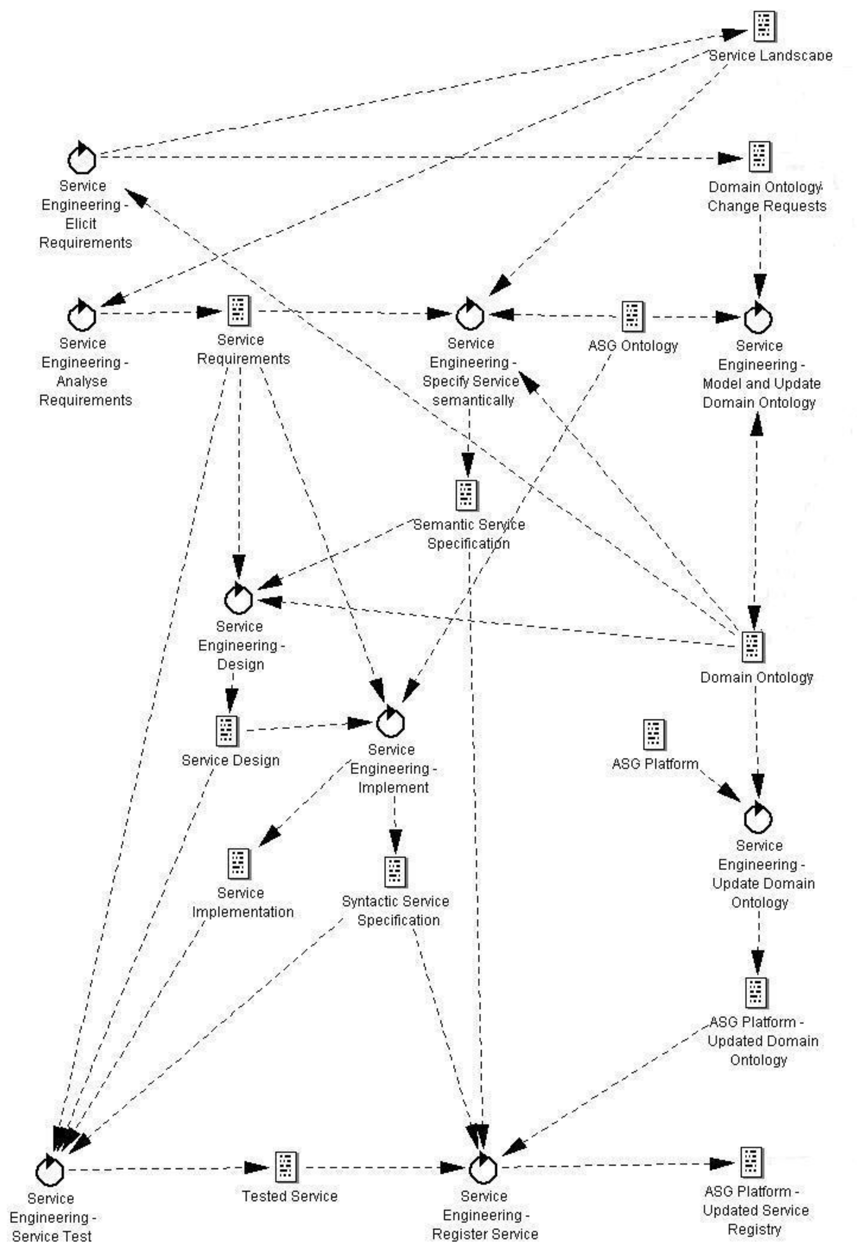
Service Landscape
Service Engineering - Elicit Requirements
Domain Ontology Change Requests
Service Engineering - Analyse Requirements
Service Requirements
Service Engineering - Specify Service semantically
ASG Ontology
Service Engineering - Model and Update Domain Ontology
Semantic Service Specification
Service Engineering - Design
Domain Ontology
Service Design
Service Engineering - Implement
ASG Platform
Service Engineering - Update Domain Ontology
Service Implementation
Syntactic Service Specification
ASG Platform - Updated Domain Ontology
Service Engineering - Service Test
Tested Service
Service Engineering - Register Service
ASG Platform - Updated Service Registry

Figure 13: Engineering Process : Service Semantic not covered in Existing Domain Ontology

4.1.3 Service Reengineering

Description of the Situation

If some functionality has been developed by a service provider that wants to provide this functionality on an ASG platform, the functionality can be reengineered for ASG, that is, reused and packaged as ASG compliant service. Reengineering is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form.

If the functionality is already a service, the task of service reengineering is a done by wrapping the existing service so that it can be deployed in the ASG platform. If, however, the functionality does not exist in an isolated way but is embedded in an application, first, the functionality must be extracted. This extraction must make ensure that the functionality can be reused by providing every part of the existing system that is necessary to use the functionality. Reengineering techniques like slicing, dependency analysis, or interface analysis support this task. Other reengineering techniques like feature location or clustering offer the potential to be tailored to be used to support service reengineering.

Approach

The goal of service reengineering is to take some existing functionality and prepare it in a way that a service results that can be deployed on an ASG platform. If the functionality is embedded in an existing application, reengineering techniques can be used to extract the required parts of the existing application that is used during service engineering. One approach that supports such a reengineering is ADORE [41] that realizes service reengineering in three steps:

- First, existing components are identified and reverse engineered to assess their adequacy; this activity is initiated by requests coming directly from the service engineer.
- Second, based on the analysis results, the service engineer decides whether the existing component is reused or a service is developed from scratch.
- Finally, when reusing the component, necessary renovation and extension activities are kicked off to adapt the component for its use within the service.

The syntax and semantic of the resulting service have to be specified and then registered in the ASG platform. If the concepts required to describe the functionality of the service is not covered in the existing domain ontology, the domain ontology has to be updated.

4.2 Application Engineering

4.2.1 New Application Semantic covered by Existing Domain Ontology and Application based on existing ASG Services

Description of the Situation

The end service provider wants to provide a new application that runs on the ASG platform and can be used by the end service customer. The application has to be developed first. An application as described in section 1.3 enacts a specified application flow. The functionality of the flow is supported by services that are available in the ASG platform.

Approach

The application engineer of the end service provider has to develop the applications according to the requirements of the end service customer. The application also has to be tested on the ASG platform. It has to be verified if all required ASG services are invoked correctly and if the services work as specified. Figure 14 illustrates the tailoring of ASG according to this situation.

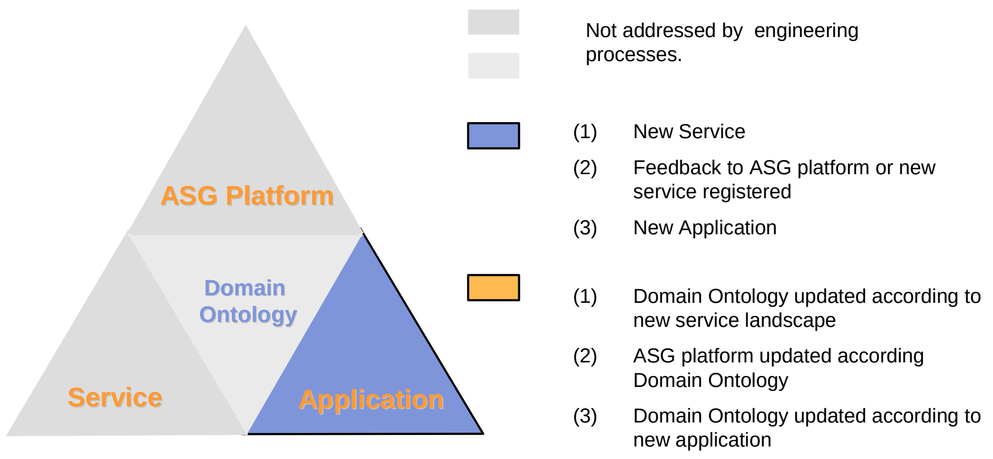
Not addressed by engineering processes.
(1) New Service
(2) Feedback to ASG platform or new service registered
(3) New Application
(1) Domain Ontology updated according to new service landscape
(2) ASG platform updated according Domain Ontology
(3) Domain Ontology updated according to new application
ASG Platform
Domain Ontology
Service
Application

Figure 14: Approach: New Application Semantic covered by Existing Domain Ontology and Application based on existing ASG Services

Figure 15 presents the engineering activities required to develop and test the new application.

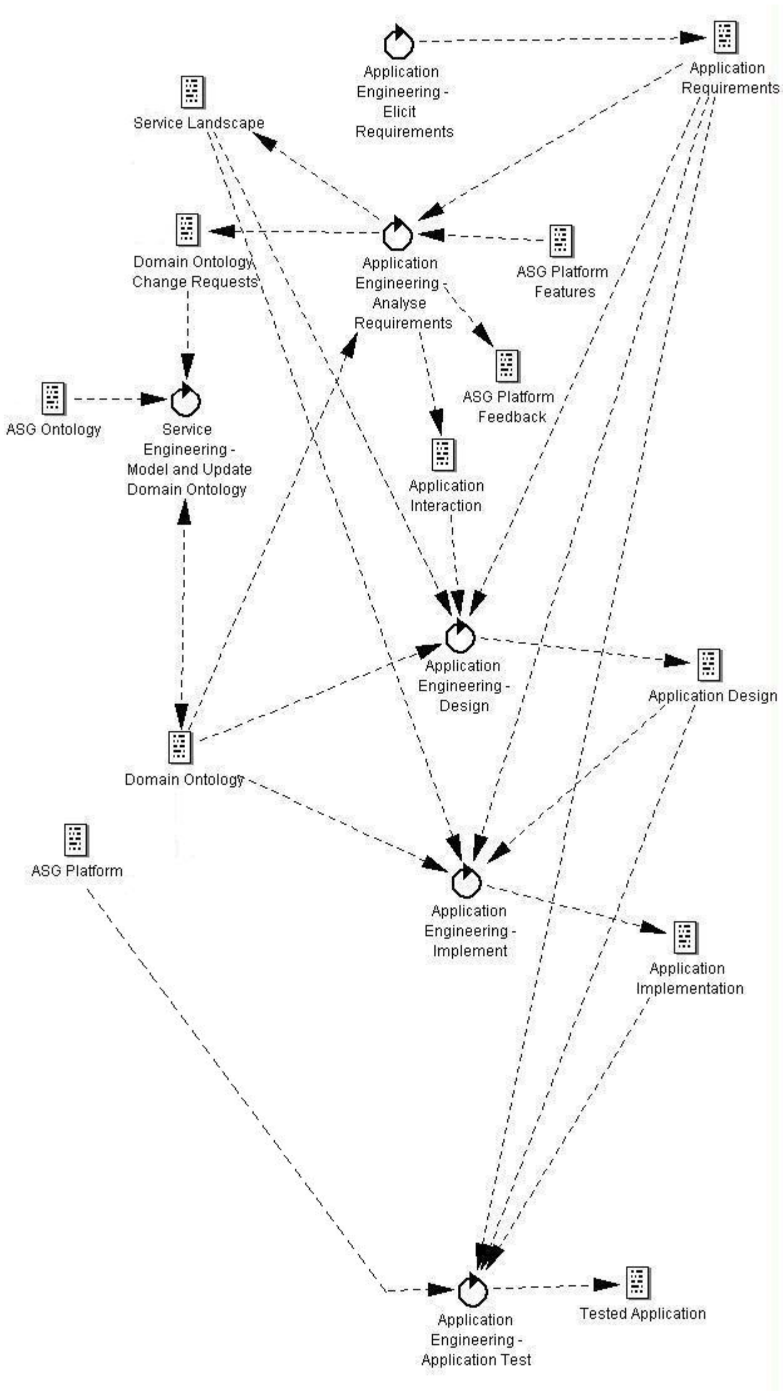
Application Engineering - Elicit Requirements
Application Requirements
Service Landscape
Domain Ontology Change Requests
Application Engineering - Analyse Requirements
ASG Platform Features
ASG Platform Feedback
ASG Ontology
Service Engineering - Model and Update Domain Ontology
Application Interaction
Application Engineering - Design
Application Design
Domain Ontology
ASG Platform
Application Engineering - Implement
Application Implementation
Application Engineering - Application Test
Tested Application

Figure 15: Engineering Process: New Application Semantic covered by Existing Domain Ontology and Application based on existing ASG Services

4.2.2 New Application Semantic covered by Existing Domain Ontology and Application based on New Services

Description of the Situation

The end service provider wants to provide a new application that runs on the ASG platform and can be used by the end service customer. The application has to be developed first. An application as described in section 1.3 enacts a specified application flow. The functionality of the flow is supported by services that are not yet available in the ASG platform. Thus, the end service provider has to develop the required services by himself or has to engage a service provider for the service development. The concepts required to describe the service are covered in the existing domain ontology (see section 4.1.1).

Approach

Thus, an application engineer of the end service provider has to develop the new application according to the requirements of the end service customer. The services required for the applications have to be identified together with the service engineer. Further, the application engineering and domain engineer have to analyze if the existing domain ontology covers the concepts to describe the functionality of the required services. It is the case in this scenario. Thus, the service engineer has to develop and register the required services as described in section 4.1.1. Subsequently, the application also has to be tested on the ASG platform. It has to be verified if all required ASG services are invoked correctly and if the services work as specified. Figure 16 illustrates the tailoring of ASG according to this scenario.

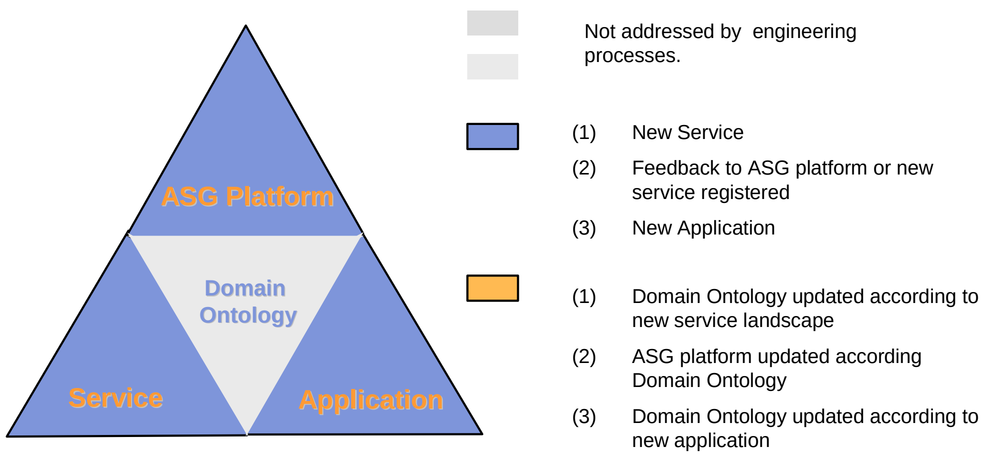
ASG Platform
Domain Ontology
Service
Application
Not addressed by engineering processes.
(1) New Service
(2) Feedback to ASG platform or new service registered
(3) New Application
(1) Domain Ontology updated according to new service landscape
(2) ASG platform updated according Domain Ontology
(3) Domain Ontology updated according to new application

Figure 16: Approach: New Application Semantic covered by Existing Domain Ontology and Application based on New Services

Figure 17 shows the engineering activities that have to be performed to develop the new application and its required services.

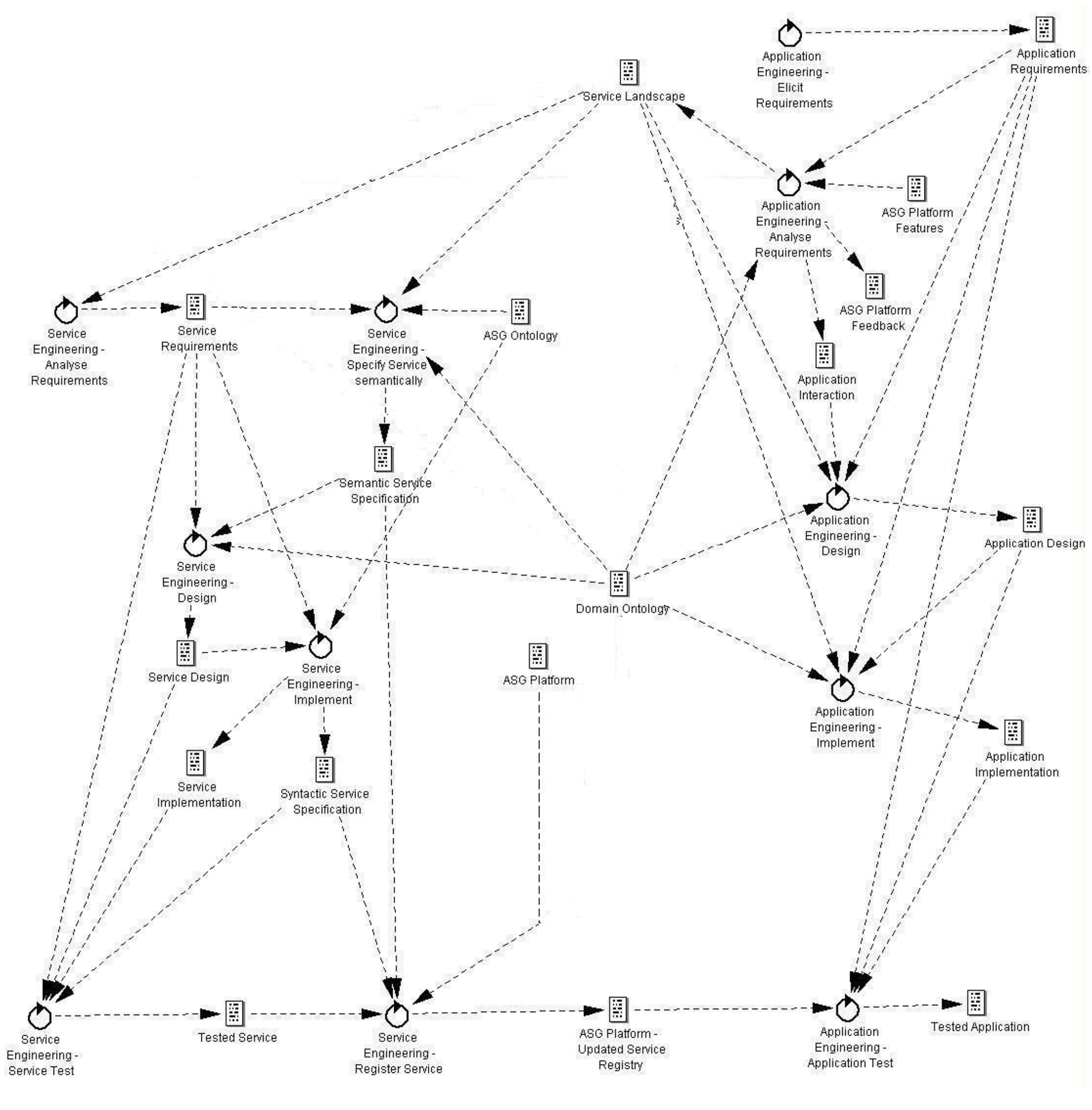
Application Engineering - Elicit Requirements
Application Requirements
Service Landscape
Application Engineering - Analyse Requirements
ASG Platform Features
ASG Platform Feedback
Application Interaction
Service Engineering - Analyse Requirements
Service Requirements
Service Engineering - Specify Service semantically
ASG Ontology
Semantic Service Specification
Service Engineering - Design
Application Engineering - Design
Application Design
Domain Ontology
Service Design
Service Engineering - Implement
ASG Platform
Application Engineering - Implement
Application Implementation
Service Implementation
Syntactic Service Specification
Service Engineering - Service Test
Tested Service
Service Engineering - Register Service
ASG Platform - Updated Service Registry
Application Engineering - Application Test
Tested Application

Figure 17; Engineering Process: New Application Semantic covered by Existing Domain Ontology and Application based on New Services

4.2.3 New Application Semantic not covered in Existing Domain Ontology and Application based on existing ASG Services

Description of the Situation

The end service provider wants to provide a new application that runs on the ASG platform and can be used by the end service customer. The application has to be developed first. An application as described in section 1.3 enacts a specified application flow. The functionality of the flow is supported by services that are already available in the ASG platform. As mentioned already the application defines specific goals so that the ASG platform can offer the appropriate services. In this scenario the concepts used in the application to specify the goal are different to the concepts specified in the domain ontology. Thus, the domain ontology has to be updated by integrating mediation, for example.

Approach

Thus, an application engineer of the end service provider has to develop the new application according to the requirements of the end service customer. During the requirements analysis the application engineer specifies the required services and recognizes that these services are already available. Further, he analyzes together with the domain engineer the changes required in the domain ontology considering the existing domain ontology and the concepts used in the application. The domain engineer updates the domain ontology and includes the new ontology in the ASG platform. Concluding the application has to be tested on the ASG platform. It has to be verified if all required ASG services are invoked correctly and if the services and the domain ontology work as specified. Figure 18 illustrates the tailoring of ASG according to this scenario.

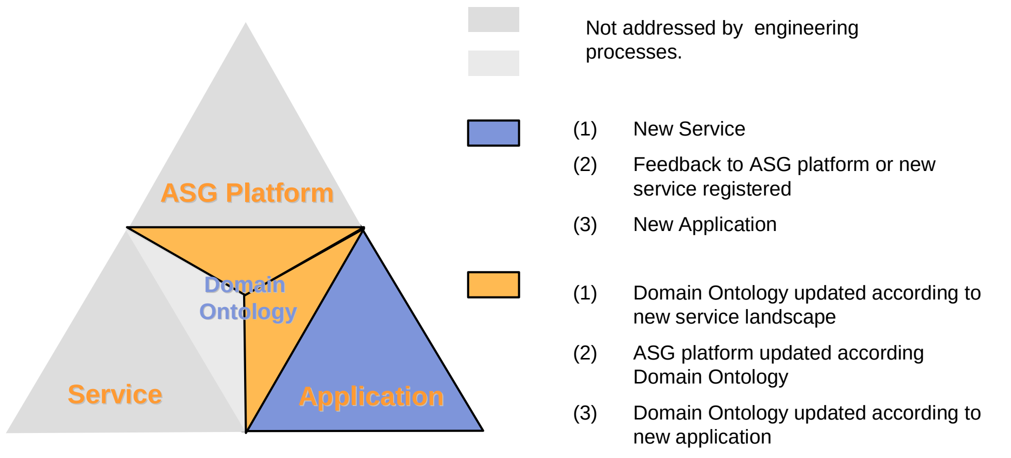
Not addressed by engineering processes.
(1) New Service
(2) Feedback to ASG platform or new service registered
(3) New Application
(1) Domain Ontology updated according to new service landscape
(2) ASG platform updated according Domain Ontology
(3) Domain Ontology updated according to new application
ASG Platform
Domain Ontology
Service
Application

Figure 18; Approach: New Application Semantic not covered in Existing Domain Ontology and Application based on existing ASG Services

Figure 19 shows the engineering activities that have to be performed to develop the new application and to update the domain ontology.

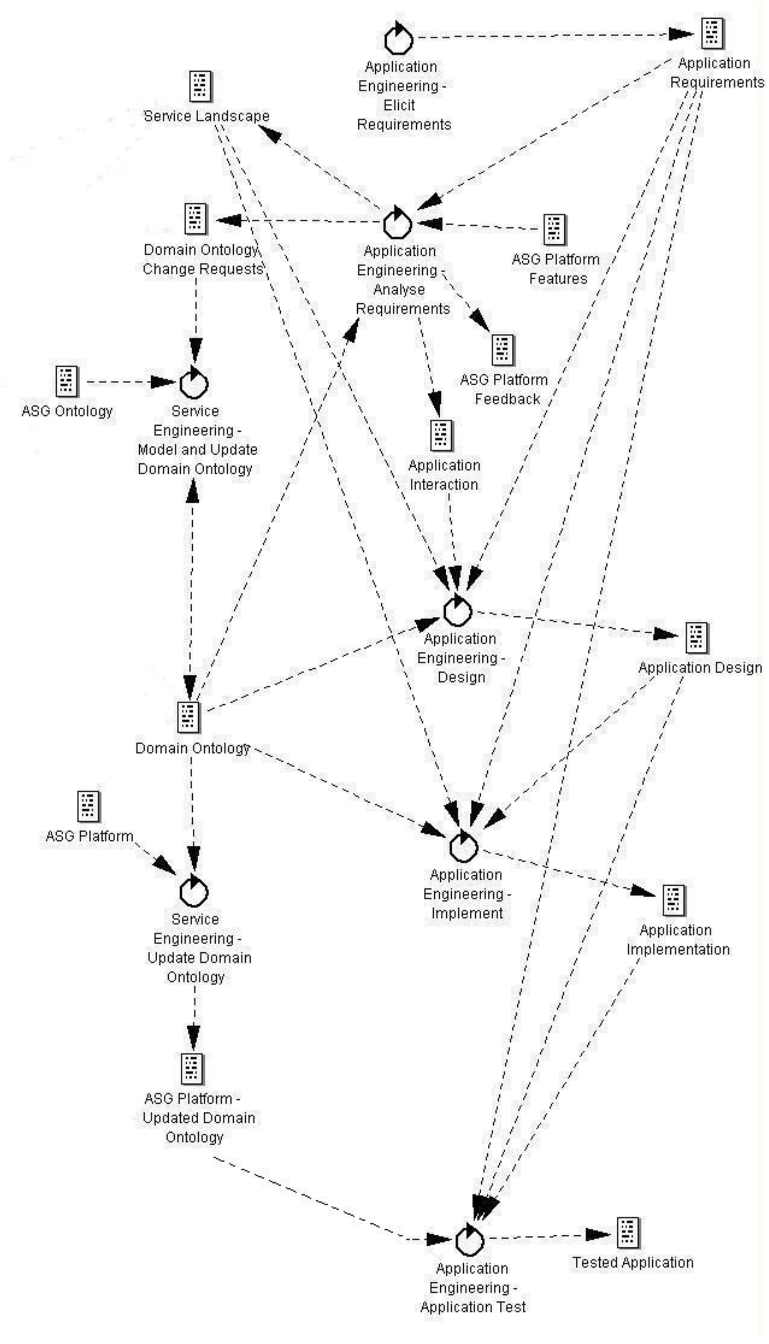
Application Engineering - Elicit Requirements
Application Requirements
Service Landscape
Domain Ontology Change Requests
Application Engineering - Analyse Requirements
ASG Platform Features
ASG Platform Feedback
ASG Ontology
Service Engineering - Model and Update Domain Ontology
Application Interaction
Application Engineering - Design
Application Design
Domain Ontology
ASG Platform
Application Engineering - Implement
Application Implementation
Service Engineering - Update Domain Ontology
ASG Platform - Updated Domain Ontology
Application Engineering - Application Test
Tested Application

Figure 19: Engineering Process: New Application Semantic not covered in Existing Domain Ontology and Application based on existing ASG Services

4.2.4 New Application Semantic not covered by Existing Domain Ontology and Application based on New Services

Description of the Situation

The end service provider wants to provide a new application that runs on the ASG platform and can be used by the end service customer. The application has to be developed first. An application as described in section 1.3 enacts a specified application flow. The functionality of the flow is supported by services that are not yet available in the ASG platform. Thus, the end service provider has to develop the required services by himself or has to engage a service provider for the service development. The concepts required to describe the service are not covered in the existing domain ontology (see section 1.1.1).

Approach

Thus, an application engineer of the end service provider has to develop the new application according to the requirements of the end service customer. During the requirements analysis the application engineer specifies the required services and recognizes that these services are not yet available in the ASG platform. Further, he analyzes together with the domain engineer the changes required in the domain ontology considering the existing domain ontology and the concepts needed to specify the functionality of the required services semantically. The domain engineer updates the domain ontology and includes the new ontology in the ASG platform. The service engineer develops the new services according to the defined requirements, specifies the semantic and syntax of the services and subsequently register them at the ASG platform. Concluding the application has to be tested on the ASG platform. It has to be verified if all required ASG services are invoked correctly and if the services and the domain ontology work as specified. Figure 20 illustrates the tailoring of ASG according to this scenario.

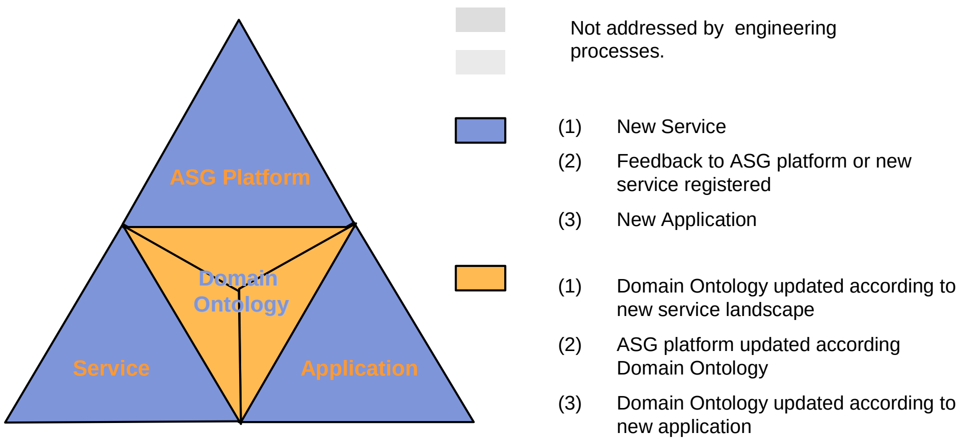
ASG Platform
Domain Ontology
Service
Application
Not addressed by engineering processes.
(1) New Service
(2) Feedback to ASG platform or new service registered
(3) New Application
(1) Domain Ontology updated according to new service landscape
(2) ASG platform updated according Domain Ontology
(3) Domain Ontology updated according to new application

Figure 20: Approach: New Application Semantic not covered by Existing Domain Ontology and Application based on New Services

Figure 21 shows the engineering activities that have to be performed to develop the new application, develop the required services and to update the domain ontology.

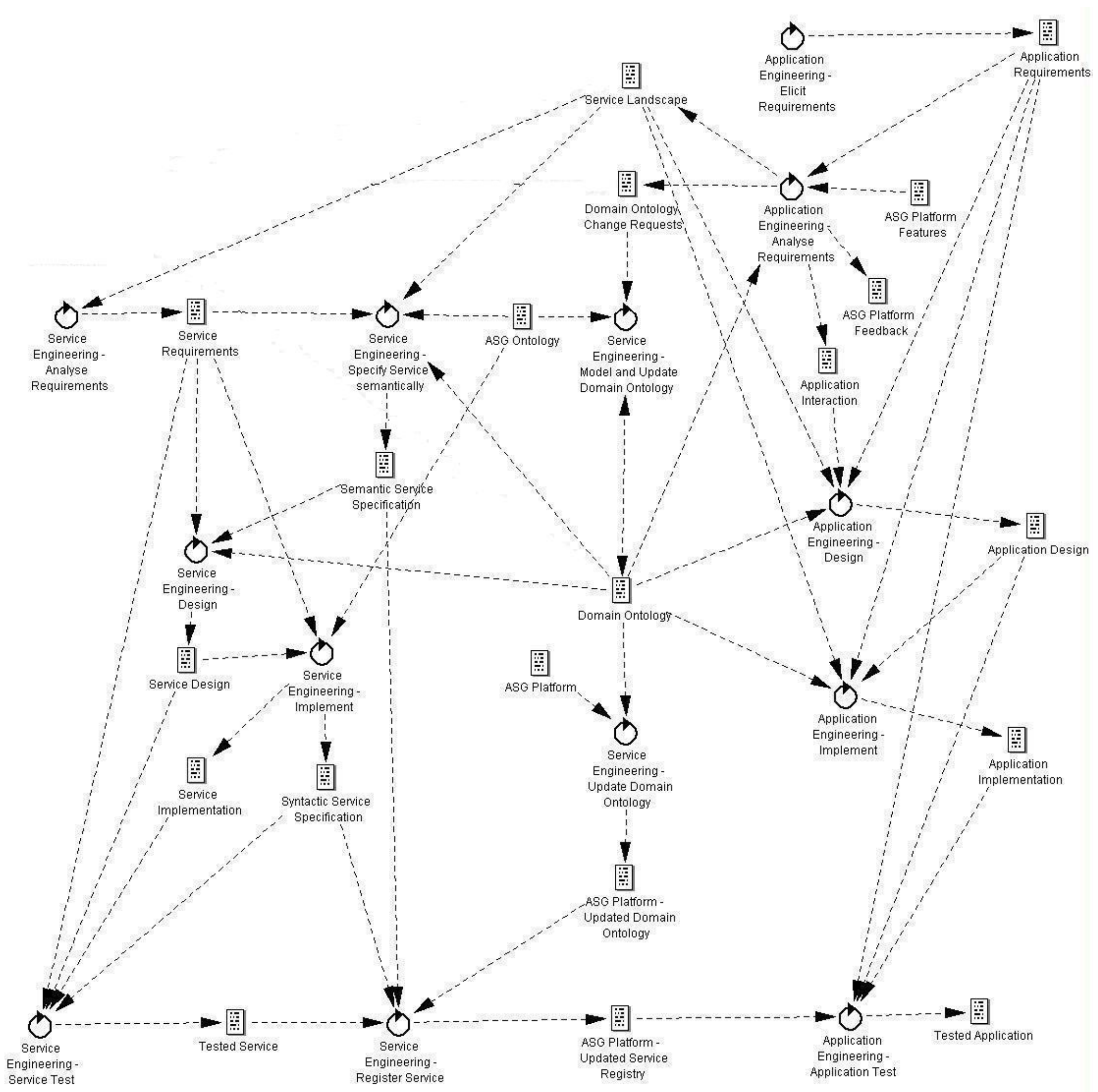
Application Engineering - Elicit Requirements
Application Requirements
Service Landscape
Domain Ontology Change Requests
Application Engineering - Analyse Requirements
ASG Platform Features
ASG Platform Feedback
Application Interaction
Service Engineering - Analyse Requirements
Service Requirements
Service Engineering - Specify Service semantically
ASG Ontology
Service Engineering - Model and Update Domain Ontology
Semantic Service Specification
Application Engineering - Design
Application Design
Service Engineering - Design
Domain Ontology
Service Design
Service Engineering - Implement
ASG Platform
Application Engineering - Implement
Application Implementation
Service Engineering - Update Domain Ontology
Service Implementation
Syntactic Service Specification
ASG Platform - Updated Domain Ontology
Service Engineering - Service Test
Tested Service
Service Engineering - Register Service
ASG Platform - Updated Service Registry
Application Engineering - Application Test
Tested Application

Figure 21: Engineering Process: New Application Semantic not covered by Existing Domain Ontology and Application based on New Services

4.2.5 Application Reengineering

Description of the Situation

In this situation, an application already exists, which is, however, not developed for an ASG platform. Consequently, the benefits of ASG can not be exploited for the application. The task in this situation is to migrate the existing application to an application that uses an existing ASG platform. This task should be performed in a way that enables as much reuse of elements of the existing application as possible, to save development effort and to reuse proven quality.

Depending on the nature of the existing application, the migration task differs. If the existing application is, for example, service-oriented, the services can be reused (after being packaged for ASG, compare 4.1.3).

Approach

To migrate an application to ASG in general the same process is followed when a new application is developed for ASG (compare 4.2.1, 4.2.2, 4.2.3, and 4.2.4). The difference, however, is that existing information and artefacts from the pre-existing application are reused. Existing artefacts might range from requirements and design to code. If the respective information about the existing application is not documented explicitly, reengineering techniques can be used to extract requirements, the architecture, or the design from the code. Reengineering techniques can also be used for clustering, that is to partition the existing system into parts that can be reused.

The services used in the migrated application can either be developed from scratch (compare 4.1.1, 4.1.2) or be reengineered from the existing application (compare 4.1.3).

5 Conclusions

In this deliverable, we presented those aspects of the ASG methodology that address engineering processes for services and applications. We described a general process for these two process areas of the overall ASG process and developed a number of scenarios that cover the different situations that can arise when an ASG platform is used. The scenarios provide customizations to the general process, so that the process fits optimally to the current situation.

The results presented in this deliverable will be further used in the course of the project. They are the basis for deliverable D6.III-7 "ASG Application and Service Engineering Approach" [42], where the processes presented here will be augmented with techniques and tools that support their enactment and combined with other results of ASG, such as the testing methodology [37]. The result is a method for developing services and service-based applications in ASG-based software development projects.

References

[1] "Reference Architecture: Requirements, Current Efforts and Design", Adaptive Services Grid Deliverable, D6.V-1 Integrated Project, Sixth Framework Programme, 2006.

[2] Weske, M., et al.: Technical Annex 1 – Adaptive Service Grid, Description of Work. Proposal no. 004617, Integrated Project, Sixth Framework Programme, 2004.

[3] "ASG Platform Development Process", Adaptive Services Grid Deliverable, D6.IV-1 Integrated Project, Sixth Framework Programme, 2006.

[4] International Organization for Standardization (ISO): ISO / IEC 12207:1995/Amd.2:2004(E): Information technology - Software life cycle processes. Amendment 2.Genf, 2004

[5] International Organization for Standardization (ISO): ISO/IEC 15504, Information Technology - Software Process Assessment - Parts 1-9. Technical Report Type 2, Genf, 1998.

[6] Ocampo, A., Boggio, D., Muench, J., Palladino, G.: Toward a Reference Process for Developing Wireless Internet Services, IEEE Transactions on Software Engineering, vol. 29, no. 12, pp. 1122-1134, December, 2003.

[7] McDermid, J.A., Rook, P.: Software Development Process Models, Software Engineer's Reference Book, Ed., Boca Raton, FL: CRC Press, pp. 15.26 – 15.28. (1994).

[8] Karlsson, E., Taxen, L.: Incremental Development for AXE 10. ACM SIGSOFT Software Engineering Notes, vol. 22, No. 6 (1997).

[9] Taylor, M.J., McWilliam, J., Forsyth, H., Wade, S.: Methodologies and Website Development: A Survey of Practice. Information and Software Technology, vol. 44, No. 6, pp. 381-391 (2002).

[10] Nerurkar, U.: Web User Interface Design: Forgotten Lessons. IEEE Software, vol. 18, No. 6, pp. 69-71 (2001).

[11] Karlsson, E.: A Construction Planning Process. Q-Labs, LD/QLS 96:0381, Lund Sweden (1999).

[12] Carlshamare, P.: Release Planning in Market-Driven Software Product Development: Provoking and Understanding. Require-ments Engineering, No. 7, pp. 139-151, (2002)

[13] Maurer, F., Martel, S.: Rapid Development for Web-Based Appli-cations. IEEE Internet Computing, vol 6, No 1, pp. 86-90 (2002)

[14] Zettel, J., Maurer, M., Münch, J., Wong, L.: LIPE: A Lightweight Process for E-Business Startup Companies based on Extreme Programming. Proceedings of the Third International Conference on Product-Focused Software Processes Improvement (PROFES), pp. 255-270, (2001)

[15] Boehm, B.W.: Get Ready for Agile Methods, with Care, IEEE Computer, vol 35, No 1, pp. 64-69 (2002).

[16] Boehm, B.W.: A Spiral Model for Software Development and Enhancement, IEEE Computer, vol 21, No 5, pp. 61-72 (1988).

[17] Palfreyman, J. (2004): Grid Explained. IBM Global Services, 2004.

[18] Bella, F, Ocampo A.: Adaptable Process Engineering Survey. (Adaptive Services Grid Deliverable D6.III-1)

[19] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke: Grid Services for Distributed System Integration. IEEE Computer. June 2002 (Vol. 35, No. 6) pp. 37-46.

[20] O. Zimmermann, P. Krogdahl, C. Gee: Elements of Service-oriented Analysis and Design: An interdisciplinary approach for SOA projects, Available at http://www- 106.ibm.com/developerworks/Webservices/library/ws-soad1/

[21] D.X. Adamopoulus, C.A. Papandreou: An integrated object-oriented approach to telecommunications service engineering, Proceedings of IFAC/IFOR/IMACS/IFIP LSS '98, Rio, Greece, pp. 834-839, 1998.

[22] D.X. Adamopolous, G. Haramis, C.A. Papandreou: Rapid prototyping of new telecommunications services: a procedural approach, Computer Communications 21, pp. 211-219, 1998.

[23] D.X. Adamopoulus, G. Pavlou, C.A. Papandreou: An integrated an systematic approach for the development of telematic services in heterogeneous distributed platforms, Computer Communications 24, pp. 394-415, 2001.

[24] M. Adler, Component business modeling - mapping the way in insurance, Available at http://www.ibm.com/industries/financialservices/doc/content/news/newsletter/1061216103.html

[25] K. Zachos, X. Zhu, N.A.M. Maiden, S.V. Jones. Seamlessly Integrating Service Discovery into RUP Requirements Processes', in Proceedings of SOSE - Service-Oriented Software Engineering, workshop at ICSE06, 28th International Conference on Software Engineering, Shanghai, China, May 2006.

[26] A. Kozlenkov V. Fasoulas F. Sanchez G. Spanoudakis A. Zisman. A Framework for Architecture-driven Service Discovery, International Workshop on Service Oriented Software Engineering (IW-SOSE'06), ICSE 2006, China, May 2006.

[27] M. Reapple: IT-Ballet. Vier Process Engines im Vergleich (Comparison of four process engines), iX- Magazin für Profesionelle Informationstechnik. 2004.

[28] W. Liu, G. Goldzmidt, J. Joseph: On demand business process life cycle, Part 5: Workflow development, deployment, and testing, Available at: http://www- 128.ibm.com/developerworks/library/ws-odbp5/?ca=dnt-64

[29] E. Motta, J. Dominguez, L. Cabral., M. Gaspari: IRS-II: A Framework and Infrastructure for Semantic Web Services. In: Fensel, D., Sycara, K., Mylopoulos, J. volume eds.): The SemanticWeb - ISWC 2003. Lecture Notes in Computer Science, Vol. 870. Springer- Verlag, Heidelberg (2003) 306–318.

[30] OWL Services Coalition (2003): OWL-S: Semantic Markup for Web Services, (http://www.daml.org/services/owl-s/1.0/), viewed 15 Feb 2005.

[31] D. Roman, H. Lausen, and U. Keller: Web Services Modeling Ontology Standard, WSMO Working Draft v02, 2004.

[32] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A.,   Feier, C., Bussler, C., Fensel, D.: Web Service Modeling Ontology. Applied Ontology. Vol 1, No. 1, pp. 77-106, 2005.

[33] A. A. Patil, S. A. Oundhakar, A. P. Sheth, K. Verma: Semantic Web Services: Meteor-S Web Service annotation framework, Proceedings of the 13th WWW conference, May 2004.

[34] U. Becker-Kornstaedt, J. Münch; H. Neu; A. Ocampo; J. Zettel:SPEARMINT TM 6. User Manual IESE-Report, 031.02/E Fraunhofer IESE 2002.

[35] “Requirements Engineering for ASG Applications”, Adaptive Services Grid Deliverable, D6.II-1 Integrated Project, Sixth Framework Programme, 2006.

[36] Paech, Barbara; Kohler, Kirstin: “Task-Driven Requirements in Object-Oriented Development”, Kluwer Academic Publishers, 2004.

[37] “Testing Methodology for ASG Applications and Services”, Adaptive Services Grid Deliverable, D6.III-5 Integrated Project, Sixth Framework Programme, 2006.

[38] http://cmap.ihmc.us

[39] http://www.wsmostudio.org/

[40] http://www.magicdraw.com

[41] Knodel, Jens; Girard, Jean-Francois, “Request-driven Reverse Engineering for Product Lines”, Quelle: Workshop Reengineering Prozesse (RePro 2004). Fallstudien, Methoden, Vorgehen, Werkzeuge - Proceeding. (2004)

[42] “ASG Application and Service Engineering Approach”, Adaptive Services Grid Deliverable, D6.III-5 Integrated Project, Sixth Framework Programme, 2006.

Project Consortium Information

Partner	Acronym	Contact
University of Potsdam, Germany	Universität Potsdam	Dr. Regina Gerber Universitaet Potsdam Am Neuen Palais 10 D-14469 POTSDAM Germany Email: rgerber@rz.uni-potsdam.de Tel: +49-331-9771080
University of Leipzig, Germany	UNIVERSITÄT LEIPZIG Faculty of Economics and Management **Information Systems Institute**	Prof. Bogdan Franczyk Universitaet Leipzig Ritterstrasse 26 D-04109 LEIPZIG Germany Email: franczyk@wifa.uni-leipzig.de Tel: +49.341-33720
University of Innsbruck, Austria	universität innsbruck	Prof. Dieter Fensel Institute of computer science University of Innsbruck Technikerstr. 25 A-6020 Innsbruck Austria Email: dieter.fensel@deri.org Tel: +43-512-5076488
Fraunhofer IESE, Germany	IESE **Fraunhofer** Institut Experimentelles Software Engineering	Dr. Dirk Muthig Fraunhofer-Gesellschaft zur Foerderung der angewandten Forschung e. V. Hansastrasse 27C, D-80686 MUENCHEN Germany Email: muthig@iese.fhg.de Tel: +49-6301-707161
DaimlerChrysler Research, Germany	DaimlerChrysler	DI Jens Weiland DaimlerChrysler AG Postfach 2360 D-89013 Ulm Germany Email: jens.weiland@daimlerchrysler.com Tel: +49-731-5052404

HPI at University Potsdam, Germany		Hasso-Plattner-Institut fuer Softwaresystem-technik gGMBH Prof.-Dr.-Helmert-Strasse 2-3 D-14482 POTSDAM Germany Prof. Mathias Weske Email: Mathias.Weske@hpi.uni-potsdam.de Tel: +49-331-5509191 Prof. Andreas Polze Email: andreas.polze@hpi.uni-potsdam.de Tel: +49 331 5509 231
NUIG, Ireland		Prof. Christoph Bussler National University of Ireland Science and Engineering Technology Building Galway Ireland Email: chris.bussler@deri.ie Tel: +353-87-6826940
Swinburne University, Australia		Prof. Ryszard Kowalczyk Swinburne University of Technology PO Box -218 AUS-3122 HAWTHORN Australia Email: rkowalczyk@it.swin.edu.au Tel: +61-39-2145834
Thueringer Anwendungszent-rum fuer Software-, Informati-ons- und Kommunikations-technologie GmbH, Germany		DI Holger Krause Thueringer Anwendungszentrum fuer Soft-ware-, Informations- und Kommunikations-technologie GmbH Langewiesener Strasse 32 D-98693 ILMENAU Germany Email: Krause@transit-online.de Tel: +49-3677-845109
NIWA, Austria		DI Alexander Wahler NIWA-WEB Solutions Niederacher & Wahler OEG Kirchengasse 13/1a A-1070 VIENNA Austria Email: wahler@niwa.at Tel: +43-131-9584311

Telenor Communications II AS, Norway	telenor	Dr. Rolf. B. Haugen Telenor ASA Snaroeyveien 30 N-1331 FORNEBU Norway Email: rolf-bjorn.haugen@telenor.com Tel: +47-900-51101
Siemens AG, Germany	SIEMENS	Klaus Jank Siemens Aktiengesellschaft Wittelsbacherplatz 2 D-80333 MUENCHEN Germany Email: klaus.jank@siemens.com Tel: +49 (89) 636-50573
Rodan Systems, Poland	RODAN SYSTEMS S.A. ZARZĄDZANIE INFORMACJĄ	Mariusz MOMOTKO Rodan Systems Spolka Akcyjna Ul. Pulawska 465 PL-02-844 WARSZAWA Poland Email: Mariusz.Momotko@rodan.pl Tel: +48-58-5502024
University Jyväskylä, Finland	UNIVERSITY OF JYVÄSKYLÄ	Prof. Jari Antti VEIJALAINEN Jyvaskylan Yliopisto Seminaarinkatu 15 FI-40014 JYVASKYLA Finland Email: jari.veijalainen@titu.jyu.fi Tel: +358-14-2603021
Telekomunikacja Polska, Poland	tp	Bogdan BANSIAK Telekomunikacja Polska S.A. Ul. Obrzezna 7 PL-02-691 WARSZAWA Poland Email: Bogdan.Bansiak@telekomunikacja.pl Tel: +48-22-6995340
Marketplanet, Poland	Marketplanet	Otwarty Rynek Elektroniczny S.A. Ul. Domaniewska 41 PL-02-672 WARSZAWA Poland Email: info@marketplanet.pl Tel: +48 22 576 88 00

University of Karlsruhe, Germany		PD Dr. Steffen Staab Universitaet Karlsruhe Kaiserstrasse 12 D-76131 KARLSRUHE Germany Email: staab@aifb.uni-karlsruhe.de Tel: +49-721-6084751
ASTEC Sp. z o.o., Poland		Janusz MICHALEWICZ ASTEC SP. Z O.O. Ul. Piaskowa 14 PL-65-209 ZIELONA GORA Poland Email: J.Michalewicz@astec.com.pl Tel: +48-68-3298031
The Poznan University of Economics, Poland		Prof. Witold ABRAMOWICZ Akademia Ekonomiczna W Poznaniu Al. Niepodleglosci 10 PL-60-967 POZNAN Poland Email: W.Abramowicz@kie.ae.poznan.pl Tel: +48-61-8569333
FH Furtwangen, Germany		Prof. Ulf Schreier University of Applied Sciences Furtwangen Rogert-Gerwig-Platz 1 D-78120 FURTWANGEN Germany Email: schreier@fh-furtwangen.de Tel: +49-7723-920153
Polska Telefonia Cyfrowa, Poland		Longin BRZEZINSKI Polska Telefonia Cyfrowa SP. Z O.O. Al. Jerozolimskie 181 PL-02-222 WARZAWA Poland Email: lbrzezinski@era.pl Tel: +48-22-4135808

Document Information

Title:	ASG Development Process – Application and Service Engineering Deliverable D6.III-2
Date:	June 29, 2006
Report:	IESE-134.06/E
Status:	Final
Distribution:	Public