ASG Platform Development Process
Deliverable D6.IV-1

Authors:
Fabio Bella,
Theresa Lehner,
Alexis Ocampo

Abstract

This deliverable presents the first version of the process for developing the ASG platform. The document also describes the methodology applied to formalize and validate the process.

The development process description presented is intended as a reference for planning, performing, and monitoring the ASG development activities. Therefore, the deliverable is intended for all parties involved in the development of the platform and for those people who need more insight into the various interrelated development activities.

Keywords: ASG, Software Process Management

Executive Summary

This deliverable presents the first version of the process for developing the ASG platform. The document also describes the methodology applied to formalize and validate the process.

Major contributions included in the deliverable are an approach to software process management based on international standards, a conceptual model of the entities used to describe the ASG development process, a description of the parts of the ASG development process that are concerned with the engineering of the platform, and a discussion of the process as applied up to milestone M18.

This deliverable does not directly address any of the ASG key features but rather the process aimed at developing the platform that provides them all. The key features are briefly introduced in the document as factors that influence the content and structure of the process.

The development process description presented is intended as a reference for planning, performing, and monitoring the ASG development activities. Therefore, the deliverable is intended for all parties involved in the development of the platform and for those people who need more insight into the various interrelated development activities.

Table of Contents

1 Introduction

One of the most significant results of the ASG project is a software infrastructure that aims at enabling design, implementation, and use of applications based on adaptive services, namely the ASG platform. This deliverable deals with the process applied in the project to develop the platform and the activities performed to define, establish, and evaluate the development process.

In this section, the need for a work component focusing on process issues is stated and the main contribution of the deliverable is introduced. The ideas behind adaptive service provisioning as addressed within the scope of the project are presented in order to understand some of the characteristics of the intended platform that influence the structure and content of the development process more than others. The relationships to other process-related ASG deliverables are highlighted by explaining the different aspects of the development process addressed. Finally, the structure of the deliverable is presented.

1.1 Motivation and Contribution

Activities such as requirements analysis, design, implementation, integration, and testing are performed in the majority of the projects aimed at developing software and international standards such as the ISO/IEC 12207 "Software Lifecycle Processes" [9] represent a good starting point for the description of software processes. However, each organization aimed at developing software has its own peculiarities that make the definition of a specific development process unavoidable. This becomes particularly true with increasing organization size. Development activities are performed, for instance, within the ASG project by several teams from different companies, universities, and research institutes. Development teams range from two-person teams consisting of a PhD student and a master student to ten professional programmers. Development teams are not collocated and team members speak different native languages. For all these reasons, there is a strong need in the project for a common terminology for process-related terms such as activity, artefact, development cycle, etc., but also for terms related to the contents of the process, such as names of specific activities and artefacts.

The ASG process terminology (also called process architecture, meta-model, or conceptual model) is presented in section 4.1.

The ASG development process and, in particular, the part of the process that aims at engineering the ASG platform is described in sections 4.2 and 4.3. The

1

development process description presented in this deliverable (and also available on the project server) is intended as a reference for planning, performing, and monitoring the ASG development activities. The ASG development process is defined as an iterative, incremental process that is driven by the development of demonstrators (so-called scenarios).

An explicit description of the development process enables systematic process analysis; Section 3.2 presents the approach followed in the project to process definition, establishment, and improvement.

Since the ASG project is a research project rather than a software development project, lessons learned during development turn out to be at least as much important as the software developed. A post-mortem analysis of the development activities performed is crucial for identifying and classifying such lessons learned. The analysis of the development process on the basis of explicit process descriptions may simplify such a post-mortem analysis.

1.2   Adaptive Service Provisioning

This section introduces the main ideas behind adaptive service provisioning in order to understand the functionality and characteristics of the platform to be developed. One of the major goals of the ASG project is to develop an open platform for adaptive and flexible service discovery, creation, composition and enactment.

Figure 1:   ASG Service Delivery Lifecycle[1]

In particular, the following research challenges are addressed by the project:

- "Semantic specification of services including functional and non-functional properties of services

- Dynamic service composition based on semantic service specifications

- Automatic negotiation of service level agreements (SLA) based on user-specified quality of service (QoS) parameters

- Easy integration of external standardized services incl. registration and deployment

- Adaptive service enactment including monitoring of Service Level Agreement (SLA) fulfilment, replanning and renegotiation as well as service profiling"[1]

The functionality to be provided by the intended architecture is shown in Figure 1[1], which depicts the ASG Service Delivery Lifecycle. At the beginning of the cycle, the current situation and the goal of an end service consumer are stated either directly or through a dedicated application. During the planning sub-cycle, services are discovered and composed to obtain a process that lets the user achieve his/her goal. During the agreement sub-cycle, Quality of Service parameters are used to negotiate the service implementations that best fit to the user's non-functional constraints. During the enactment sub-cycle, the service implementations already negotiated for the composition are invoked. New services can be registered at runtime. The respective service specification is enhanced with semantic meta-data during the integration of the service into the platform.

---

[1] From the official project flyer available at the project site https://asg-platform.org/cgi-bin/twiki/view/Public/WebHome

Vetere et al. [25] define a semantic layer as the "…methodologies, artefacts, and techniques aimed at the correct interpretation and implementation of service descriptions…". "In a service-oriented environment, the semantic layer ensures that data embedded within messages are interpreted by providers and consumers as representing the same concepts, relations, or entities in a suitable abstraction of the real world." Concerning the possible approach to semantic interoperability in service-oriented architectures, four different models are proposed [25]. The models are classified based on two fundamental dimensions:

- Integration mappings set up – There are "…two possible ways to set up integration mappings, one in which each service schema is mapped to any other (any-to-any) and another in which each one is mapped to a single schema (any-to-one)"
- Integration logic execution – There are two possibilities of how "…integration logic is executed: in a single distinguished node (centralized) or the execution is distributed among multiple, functionality equivalent nodes (decentralized)"

The approach followed in the ASG project can be classified as an any-to-one centralized model, where input/output data are mapped to a so called domain ontology managed by means of a specialized application and single services are centrally integrated into one composed service through dynamic service composition.

1.3 Process-related ASG Deliverables

The work on the processes that aim at developing software in the ASG project focuses on several aspects addressed from different points of view. For this reason, the discussion about process-related issues is spread among several deliverables of the work component C6 "ASG Development Methodology".

The ASG Development Process is structured as a set of activities grouped into three main process groups: Platform, Application, and Service Engineering. The group *Platform Engineering* consists of activities aimed at engineering the infrastructure that supports the intended ASG functionality. *Application Engineering* groups those activities aimed at engineering an end user application by combining the services available with the features of the ASG platform. All the activities that deal with services to be integrated into the ASG platform belong to the process group *Service Engineering*.

Process-related deliverables address or refine different parts of the whole development process. Figure 2 shows an overview of the deliverables and their relationships to the main process groups.

Deliverables D6.III-7 and D6.III-8 address aspects relevant for **all process groups**.

D6.III-7 ASG Application and Service Development Approach – This deliverable condenses the results of work package D6.III together with the ASG requirements engineering technique into the ASG method for developing services and service-oriented applications.

D6.III-8 ASG Quality Modeling Approach - Non-functional requirements are, by their nature, intertwined. Through a dependency analysis, conflicting requirements can be identified and resolved. This deliverable contains an approach for eliciting and documenting conflicting non-functional requirements for ASG.

Deliverables D6.IV-1-M18, D6.IV-1-FINAL, D6.III-1 [28], D6.IV-2 [34], D6.IV-3 [35], and D6.V-1 address aspects of **platform engineering**.

D6.IV-1-M18 ASG Platform Development Process – This is the present document, which introduces the first version of the process for developing the ASG platform and describes the methodology applied to formalize and validate the process.

D6.IV-1-FINAL ASG Platform Development Process – The deliverable presents the final version of the process for developing the ASG platform. The document also includes lessons learned from developing the platform by means of the process described.

D6.III-1 Adaptable Process Engineering Survey – This M6 deliverable surveys available processes for developing solutions based on adaptive service provisioning. The deliverable represents a starting point and the basis for work concerned with engineering processes performed in the ASG project.

D6.IV-2 Tracing and Logging Concept – In this deliverable, a tracing and logging concept for the ASG platform is presented. It contains tracing and logging rules and guidelines. The report describes what to trace/log and how to analyse collected data.

D6.IV-3 Testing Methodology for Platform Code - The aim of this deliverable is to provide a testing methodology for ASG platform code. The methodology encompasses testing strategies, approaches and techniques and is an integral part of the ASG platform development process.

D6.V-1 Reference Architecture - This report summarizes the final version of the ASG reference architecture and its evolution throughout the project. The ASG architecture has a direct impact on platform development processes, since it partially reflects the structure of the teams involved in developing the platform.

Deliverables D6.III-2 [32], D6.III-5 [33], and D6.I-1 [29] address aspects relevant for **application and service engineering**.

D6.III-2 ASG Development Process – Application and Service Engineering - To use the ASG platform, application and services have to be developed. The processes for application and service engineering, with their activities, artefacts, roles, tools, and assets are specified in this deliverable.

D6.III-5 Testing Methodology for ASG Applications and Services – This deliverable contains an assisting approach for the ASG development process, a testing methodology for ASG. The methodology encompasses testing strategies, approaches, and techniques for ASG applications and services.
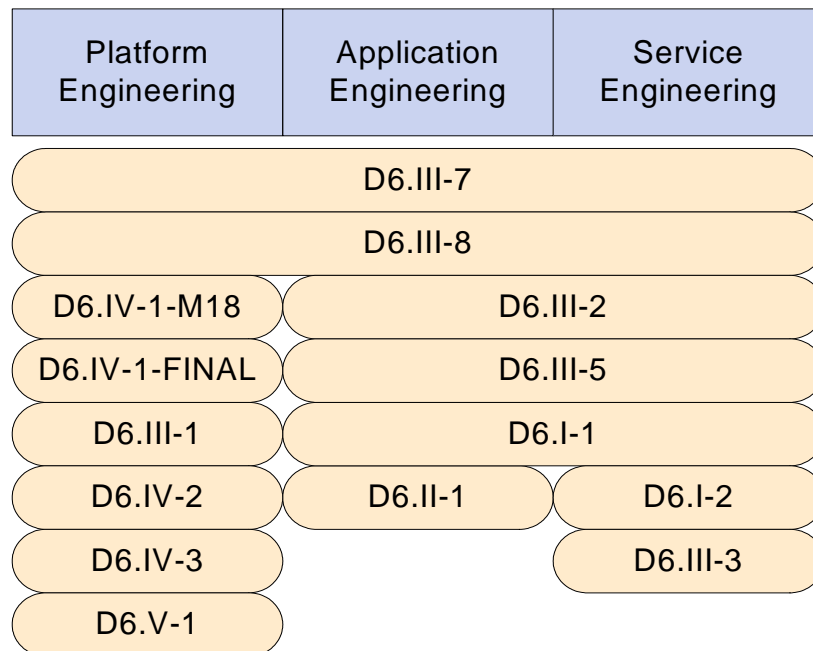
Platform Engineering	Application Engineering	Service Engineering
D6.III-7		
D6.III-8		
D6.IV-1-M18	D6.III-2	
D6.IV-1-FINAL	D6.III-5	
D6.III-1	D6.I-1	
D6.IV-2	D6.II-1	D6.I-2
D6.IV-3		D6.III-3
D6.V-1		

Figure 2:   Overview of Process-related Deliverables

D6.I-1 Requirements Specification Survey - This report describes the current state-of-the-art and state-of-the practice in service-oriented requirements specification. The survey is focused on the relationship between use cases, services, and processes to understand how services can be used to determine processes and use cases (i.e., service-based application specification). The survey is used as a basis for developing the requirements engineering method for ASG.

Deliverable D6.II-1 [31] addresses aspects relevant for **application engineering**.

D6.II-1 Case Study: Requirements Specification - This report describes a case study that applies the ASG requirements engineering method for applications. The first part documents the requirements engineering methodology for applications that are based on the ASG platform. The second part documents the results of the requirements engineering method applied on the ASG dynamic supply chain scenario.

Deliverables D6.I-2 [30], and D6.III-3 address aspects relevant for **service engineering**.

D6.I-2 Reuse-Oriented Requirements Technique – This deliverable describes a requirements engineering technique especially developed for service-oriented adaptive systems realized in the ASG context. The technique targets service providers, in particular.

D6.III-3 Performance Engineering Methodology – The performance methodology is a step by step guidance to support estimation and evaluation of the performance behaviour of ASG services.

1.4 Structure of the Deliverable

The remainder of this deliverable is structured as follows:

Section 2 "Related Work" introduces the approaches to software development that best fit the purposes of development activities performed within the scope of the ASG project. The section therefore illustrates 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.

Section 3 "Software Process Management in the ASG Project" discusses the approach followed in the project to define, establish, and evaluate the process to develop the ASG software. This section also presents the infrastructure applied to exchange process-related information and manage the process.

Section 4 "ASG Platform Development Process" represents the core of the deliverable and discusses both structure and content of the process applied to develop the platform.

In section 5, peculiarities of the platform development process are identified and discussed. Also, the question about the generalizability of the process proposed is investigated together with the issues that still remain open.

8

Finally, section 6 "Summary and Outlook" subsumes the deliverable and sketches next steps to take as well as the relationships between the work documented in this deliverable and other parts of the project.

2 Related Work

This section introduces processes, methods, and techniques available for each of the process groups addressed: Service Engineering, Platform Engineering, and Application 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 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 [27], [1], [2],[3].

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 Modeling [4] (CBM) technique. 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.

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) [27]. 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 [22], [13] 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.

2.2   Platform Engineering

ASG focuses on developing a platform that allows the automation of issues that are manually solved in service engineering such as service discovery, and composition. The ASG approach consists of adding a semantic layer to service descriptions that allows reasoners to compose, or discover automatically such services.

Similar approaches are currently evolving concerning the engineering of solutions aimed at handling services in a semantic-oriented way: IRS [16], OWL-S [19], WSMO [23], and METEOR-S [21]. 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, until now none of these initiatives provide a documented process that describes how to engineer such a platform.

Some ideas of WSMO and its language WSML have been adopted by the ASG project. The ideas have been helpful for the development of the ASG prototypes. On the other hand, ASG generates new requirements to be implemented by the leaders of the WSMO initiative.

From a very different perspective, when looking at some objectives of ASG at a higher level of abstraction, e.g., reusability of functionality, scalability, and interoperability, one can find initiatives with similar objectives such as the *enterprise architecture frameworks*. Such frameworks are domain-specific architectures that provide semi-complete applications and can be specialized to produce custom applications. Examples are the SAP® NetWeaver®, or the extinct IBM San Francisco Framework®, whose main ideas are now implemented inside IBM Websphere®. Such frameworks are composed of common objects and business processes that can be used for building applications in a given domain. Approaches for developing such basic framework components or extending the

frameworks do no differ from the traditional ones (e.g., object-oriented programming). However, a deep understanding of the framework components and business processes is needed in order to create a new application, which is a non-trivial task due to the complexity and size of the frameworks.

Due to the novelty of the domain, developing a platform such as the ASG is a task of high complexity that requires risks to be managed and minimized. Therefore, following a strict, inflexible process model like the waterfall model is not suitable for such a development project, in which organizations must react to the context in the most appropriate manner. This is only possible by means of flexible processes [18]. The spiral model, the throwaway prototype model, the incremental development model, and the Extreme Programming approach can be considered suitable starting points for the definition of a life cycle for engineering the ASG platform.

The spiral model [6] 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 costs of identifying, analyzing and maintaining risks are high and can turn out to be too expensive for small and medium companies, or, for instance, the ASG project infrastructure.

The throwaway prototype model and the incremental development model [15] were found suitable for domains such as Internet and mobile applications [12], [17] that present similar characteristics of novelty like the ASG project. In the incremental model, essential functions are provided at the beginning of a project, and then more capable versions of the system are provided according to a strategic prioritization of the requirements to be fulfilled. 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 applications [11], [8].

The Extreme Programming approach does also reflect an incremental model and has been proposed by [14] and [26] 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 [7]. In the context of the ASG this can become a critical 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.

2.3 Application Engineering

Applications based on a platform such as the ASG can go from simple Web interfaces to more sophisticated applications running on mobile devices. Whichever the case is, lack of experience dominates. Therefore as in the case of platform engineering, it is advisable to avoid or minimize risks by following life cycle models such as the spiral model [6], the throwaway prototype model, or the incremental development model [15].

Planning releases also becomes an issue especially in the context of ASG because developers are still learning. In that case, approaches from the area of requirements engineering and applied in areas like Internet or mobile applications [11], [8] are of great value. Furthermore, issues like scalability and performance have to be carefully addressed.

According to [20], 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 requirbements. [27] 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 [20], [28].

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.

3 Software Process Management in the ASG Project

This section presents the approach applied within the scope of the ASG project to manage software processes. Before the approach and the needed infrastructure are discussed in more detail, the distinction between project and process management is introduced according to the definitions provided by the international standards ISO/IEC12207 "Information Technology - Software Life Cycle Processes" [9] and 15504 "Information Technology – Software Process Assessment" [10].

3.1 Project vs. Process Management

Project and process management represent two different but strongly interrelated points of view on development: Project management deals with identifying, establishing, coordinating, and monitoring "activities, tasks, and resources necessary for a project to produce a product and/or service meeting the requirements" [10]. Process management deals with establishing "a suite of organizational processes for all software lifecycle processes as they apply to its business activities"[10].

M10 Prototype

This is the home of the M10 prototype development project. The development will follow the ASG development process.

Roadmap

Start	01.06.2005	Requirements specification finished
Sync-10.1	14.06.2005	Subsystem Analysis & Design finished
Sync-10.2	24.06.2005	Integration finished
M10 Release	30.06.2005	M10 released

Process Monitoring

Requirements

Due	01.06.2005
Responsibles	Requirements Manager, Release Manager
Status	finished

Output Artefacts	
Requirements Specification	AttractionBookingServiceSpecification
Release Plan	This document
Configuration Baseline	AttractionBookingServiceSpecification (r.31)

The requirements are defined in the AttractionBookingServiceSpecification. We will develop in M10 only what is needed to implement the service composition "Get Route Description" defined in there.

Analysis & Design

Due	03.06.2005
Responsibles	Release Manager, Architect
Status	in progress
Output	M10AnalysisAndDesign

Roles

Process Roles	
Customer	Frode Kileng, Josef Noll
Release Manager	Guido Laures
Requirements Manager	Martin Breest
Architect	Klaus Jank
Integrator	HaraldMeyer
Test Manager	

Responsibles

Support Components	
Project Management	Guido Laures
Build System	Harald Meyer
Testbed Infrastructure	Harald Böhme
Subsystems	
ASG Facade	Guido Laures
Client Application	Martin Breest
Framework	Guido Laures
C-1 Ontology and Service Sepecification	Ioan Toma
C-2 Discovery Database	Bernhard Tausch
C-2 Service Composer	Harald Meyer
C-3 Service Creation	Daniel Wiese
C-4 Mediated Replanning	Mohan Baruwal Chhetri
C-4 Negotiation Manager	Mohan Baruwal

Figure 3:   Excerpt of Release Page from the ASG Wiki

According to this distinction, process management activities include defining process goals, identifying activities and roles, helping in deploying the process, checking process conformance, defining and documenting the processes as performed, capturing process data, and maintaining process descriptions.

On the other hand, project management activities include identifying tasks, evaluating the feasibility of achieving the process, planning and allocating resources and infrastructure, implementing activities, monitoring project execution, reviewing work products and evaluating results, taking action on deviation from plan (i.e., replanning), and demonstrating successful achievement.

Figure 3 shows an excerpt of a Web page used for planning the release for M10 according to the activities defined in the process description. On the page, information about allocation of resources, main tasks, and important changes of the plan or the allocation were provided and managed.

The next section shows how process management activities are performed in ASG to support project management and, at the same time, project management information is used to manage and improve the software process.

3.2   ASG Software Process Management

As stated in section 1.1 "Motivation and Contribution", process management activities are performed in the ASG project to provide

- explicit guidance for developing the ASG platform,
- a reference taxonomy for terms related to the development of the platform,
- sound collections of document examples, templates, and guidelines,
- explicit definitions of roles, activities, artefacts, tools, and their mutual relationships, and
- suitable resources for quality management such as explicit goals for documents and quality criteria for their evaluation.

The approach defined for managing the software process is shown in Figure 4 and consists of five main phases: Initial Process Drafting, First Process Refinement, Second Process Refinement, Process Stabilization, and Post-Mortem Analysis.
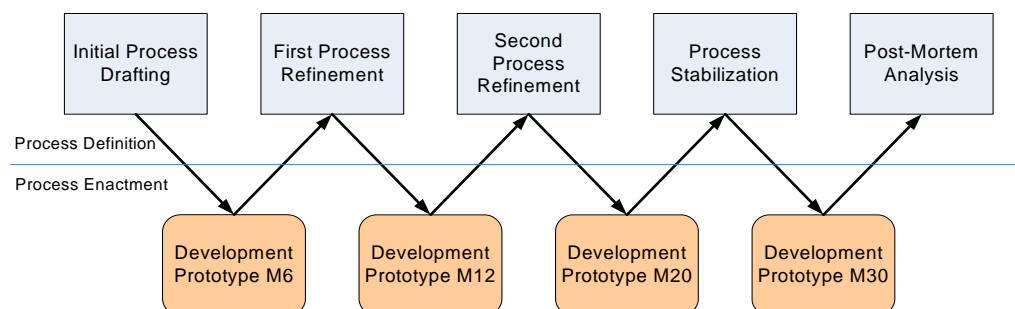
Figure 4:   ASG Approach to Software Process Management

Initial Process Drafting. During this phase, a first process draft is sketched on the basis of recognized international standards such as ISO/IEC 12207 "Information Technology - Software Life Cycle Processes" [9], and 15504 "Information Technology – Software Process Assessment" [10] and discussed with the managers of the development teams 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 prototype.

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 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 is 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.

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. In order to use as much evidence emerging from the development activities as possible, the five phases are synchronized with the major milestones defined for the project. This means that beside the first phase, which started with the project, each phase begins after a major milestone is achieved.

3.3   Process Infrastructure

A great part of the infrastructure implemented to exchange project information is also used to establish the development process. Process-related information is exchanged, therefore, over several channels consisting of regular teleconferences and meetings on the one hand, and technical solutions such as a project portal, electronic process guides, and a Web-based project management solution on the other hand.

Figure 5:   Excerpt of the ASG Development Page on the ASG Wiki

Teleconferences are held regularly with the participation of ASG members involved in development activities. Usually, the teleconferences are held once a moth. Before major milestones, the teleconferences are held twice a month or even weekly. Their main goal is to discuss the status of development activities, eventual problems and /or risks, and to make decisions about the development of demonstrators. From the point of view of process management, the teleconferences are important sources of process evidence, i.e., many discussions provide hints about how the process is enacted, which parts of it are performed without impediments and which parts need deeper analysis as a consequence of weak or even wrong process formalization.

Two different kinds of meetings play an important role for process management: Regular work-component workshops provide an insight into the degree of process establishment in the different development teams. Multi-component workshops held for a particular purpose such as the "Dynamic Supply Chain Workshop" held February 8, 2006 during the 4th ASG week in Jyväskylä offer the opportunity to enact specific process activities and validate the process description with these.
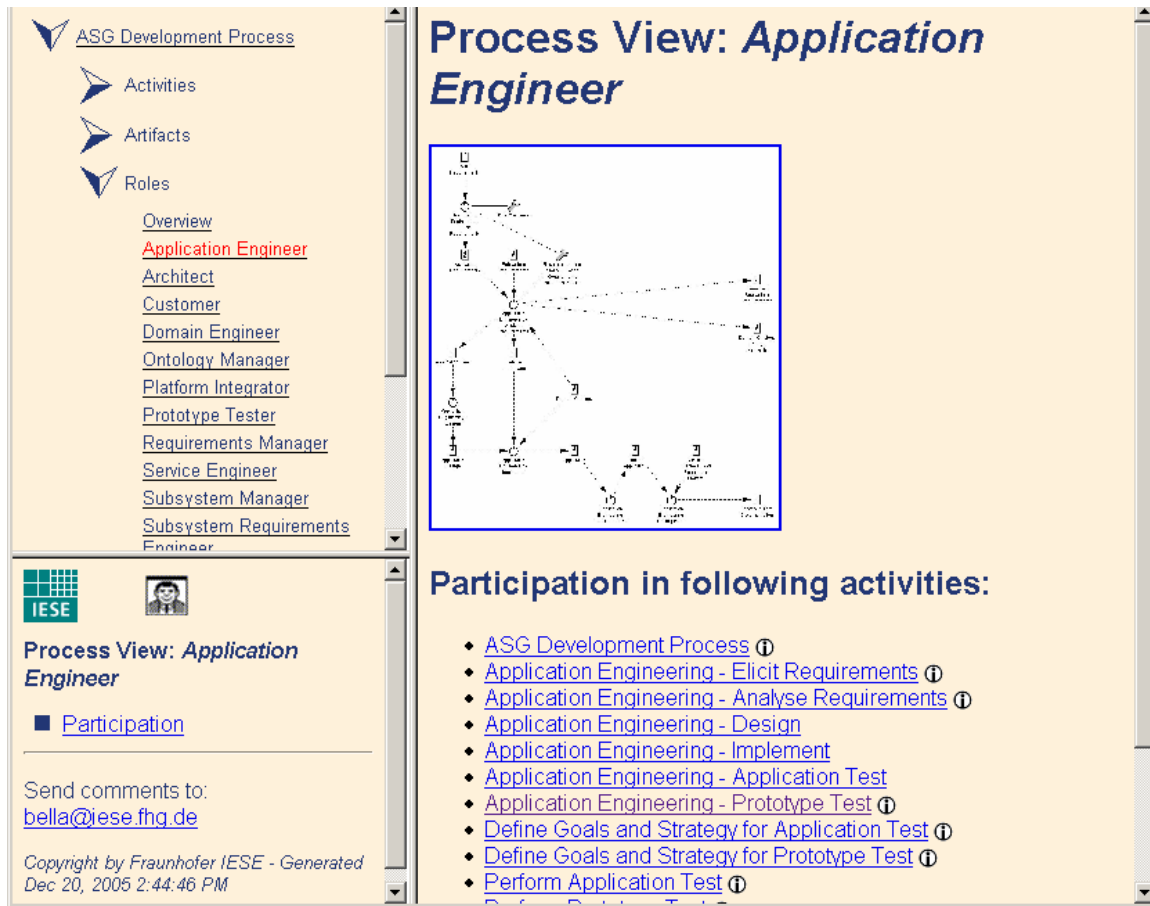
Figure 6:   Excerpt of the Electronic ASG Development Process Guide

A Wiki server, also called ASG Wiki[2], is used as project portal, that is, a Web portal to exchange ASG internal information. Figure 5 shows an excerpt of the ASG Wiki page dedicated to the ASG development process. Developers found it very useful to share available templates and descriptions of development tools through the Wiki server. Due to its importance, the location of these assets is also provided in the process description.

An Electronic Process Guide (EPG) is a process description provided as a set of interlinked Web pages [5]. An EPG created for the ASG development process can be reached from the development process page on the ASG Wiki. Figure 6 shows an excerpt of the EPG. Entities are grouped according to their type, that is, activities, artefacts, roles, and tools are grouped together. Furthermore, graphical refinements such as role-specific views are provided. One useful fea-

---

[2] The main page of the ASG Wiki is public and can be viewed at https://asg-platform.org/cgi-bin/twiki/view/Public/WebHome.

ture is the possibility to navigate through the process description by clicking on entities depicted in the graphical refinements.
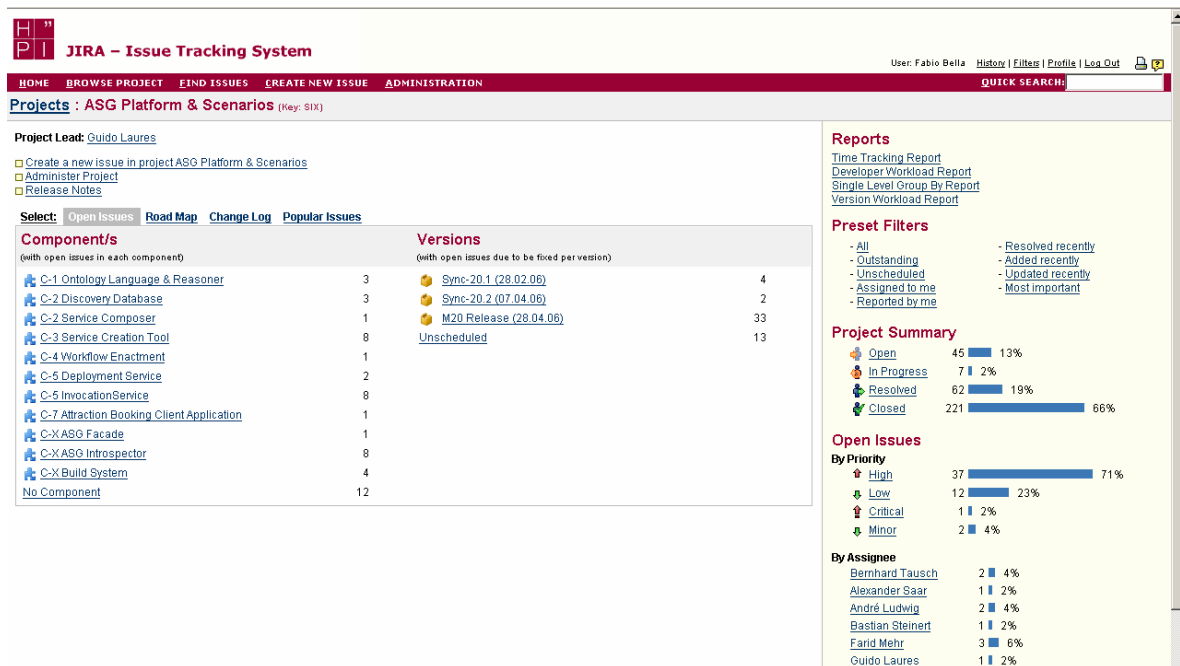

Figure 7:   Excerpt from the ASG Task Management Environment

During development, many tasks must be defined and monitored. Ideally, each task should be regarded as an instance of (part of) an activity defined in the process description. For several reasons, this cannot always be the case: a process description, for example, can not foresee every situation and task to deal with; since process descriptions are intended for human beings, they should not be too detailed to avoid unneeded overhead and frustration; it is normal that process descriptions get out of date. Development tasks can be comfortably managed with the aid of a dedicated tool. This may also help to analyse the tasks with respect to process adherence. The commercial tool JIRA (Figure 7) has been adapted to reflect the ASG development process and is used to manage the individual development cycles.

4 ASG Platform Development Process

This section is the core of the deliverable and describes the ASG development process as defined after the second process refinement (i.e., after the milestone M12). The whole development process consists of three groups of development activities, so-called *Process Groups*: Platform Engineering, Application Engineering, and Service Engineering. This deliverable focuses on the process group Platform Engineering. Application and Service Engineering are discussed in detail in the deliverable D6.III-2 "ASG Development Process – Application and Service Engineering".

The remainder of the section is structured as follows:

Section 4.1 "Process Meta-Model" introduces the concepts applied to describe the process and their mutual relationships.

Section 4.2 "ASG Development Process Overview" presents the interfaces between the process group Platform Engineering and the other groups Application and Service Engineering. The interfaces are described in terms of artefacts exchanged between related activities.

Section 4.3 "Platform Engineering" describes the process group Platform Engineering in more detail. This part of the process is aimed at engineering the infrastructure that supports (most of) the intended ASG functionality.

4.1 Process Meta-Model

The Software Process Engineering Meta-Model (SPEM) [24] is an adopted specification of the OMG that aims at standardizing how software processes are described. In the specification, the UML approach is extended to model families of related software processes. The modelling levels applied to structure object-oriented approaches are applied to the software process engineering domain. M2 is the level of the meta-model. At this level, those concepts and their relationships are described that can be used to model a software process. M1 is the level of software process models such as, for example, the Rational Unified Process (RUP) or the international standard ISO / IEC 12207. M0 is the project level: project plans and histories are examples of models within this level.

Within the scope of the ASG project, software processes are described in terms of Activities, Artefacts, Roles, Assets, and Tools. The resulting process models include both textual descriptions and diagrams that illustrate the relationships

between the entities of the model in a graphical way (e.g., workflows and role-specific views).

Figure 8 shows the process architecture applied within the project, i.e., the entities used to describe the ASG development process and their relationships. The process architecture is very close to the conceptual model presented in [24] and can be considered a subset of the SPEM. According to the figure, each activity in the process is described in terms of its purpose and tasks. Roles can be involved in different activities. Tools and assets are used to perform activities. Input artefacts are consumed in activities to produce output artefacts. Furthermore, activities may include sub-activities and artefacts may consist of sub-artefacts.
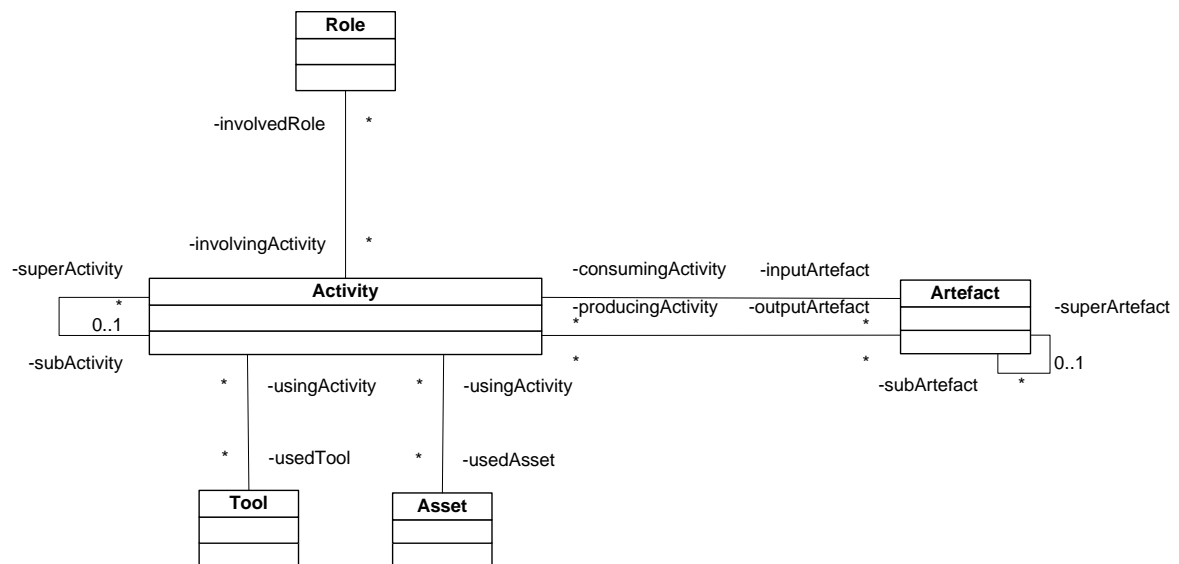
Figure 8:   Overview Process Architecture

4.2   ASG Development Process Overview

As already stated in section 1.3, the ASG Development Process is described as a set of activities grouped into three main process groups: Platform Engineering, Application Engineering, and Service Engineering.

Figure 9 shows an overview of the ASG development process with the three process groups and the artefacts exchanged. In the following, the artefacts exchanged, i.e., the interfaces between the process groups, are briefly introduced.

The process group Application Engineering produces the artefacts Prioritized Subsystem Requirements, Service Landscape, Domain Ontology Change Requests, and Prototype Test Documentation.

- *Prioritized Subsystem Requirements* is a list of requirements to be fulfilled by a subsystem, which should be implemented during a given development cycle. This artefact is used to drive the development of the platform through the implementation of applications (demonstrators).

- *Service Landscape* is a description of the services needed to run the application.

- *Domain Ontology Change Requests* are extensions of the concepts and relationships described in the domain ontology that are required to formulate problems and solutions addressed by the intended application.

- *Prototype Test Documentation* includes test cases and results for the running prototype, i.e., the end user application that calls services discovered, negotiated, composed, and eventually replanned with the aid of the ASG platform.
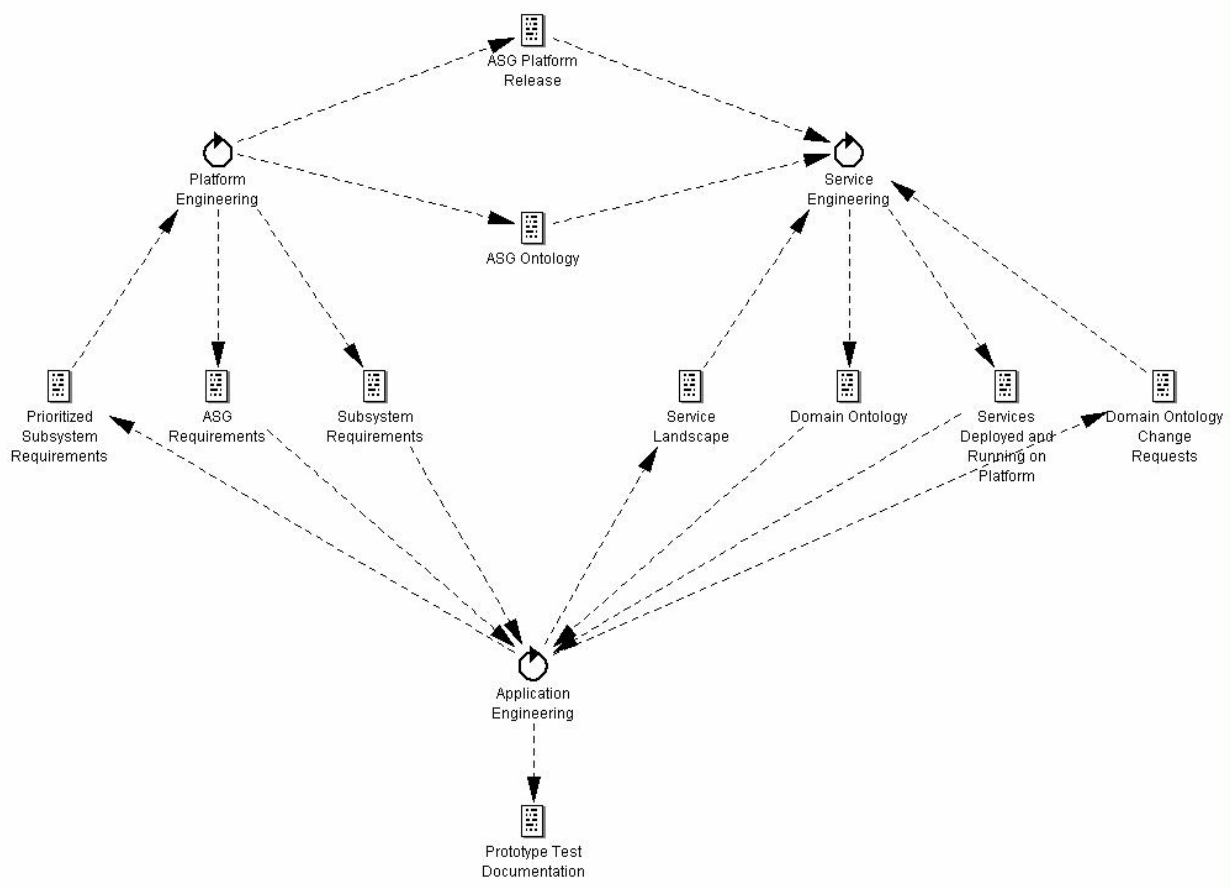
Figure 9:   Overview of the ASG Development Process

The group Platform Engineering produces the artefacts ASG Requirements, Subsystem Requirements, ASG Ontology, and ASG Platform Release.

• The artefact *ASG Requirements* is the set of features the platform has to provide.

• *Subsystem Requirements* are different sets of features to be provided by the subsystems within the platform. There is one document of this type for each subsystem.

• *ASG Ontology* is the meta-model used to specify services both semantically and syntactically and to formalize domain-specific ontologies, i.e., models of (parts of) the real world.

• An *ASG Platform Release* is a version of the platform that has already been integrated and tested. A release can be deployed and executed and serves as reference system for further development.

The group Service Engineering produces the artefacts Domain Ontology and Services Deployed and Running on Platform.

• The *Domain Ontology* is a formal model of (parts of) the real world.

• The artefact *Services Deployed and Running on Platform* represents a running ASG platform after the deployment of new services.

Platform and Service Engineering are driven in the project by Application Engineering, i.e., decisions about which platform features should be implemented first and which services should be developed and deployed on the platform are made according to the needs of the applications engineered for demonstration purposes (so called scenarios). At the beginning of each development cycle, the requirements still to be implemented in the ASG platform and its subsystems are analyzed in Application Engineering (in the figure, the artefacts ASG Requirements and Subsystem Requirements) and a list of requirements to be implemented during the cycle is produced according to the features needed for the scenarios (in the figure, the artefact *Prioritized Subsystem Requirements*).

Once the platform is engineered, the activities within the process group Platform Engineering do not need to be performed anymore. The activities aimed at engineering new applications and services make use of the platform as it is and no new platform-related requirements are elicited or implemented. Maintenance activities can be performed if required.

The overview of the ASG Development Process (see Figure 9) shows all interface artefacts for which a new version can be created during one development cycle. To synchronize the work to be performed within the different process groups, specific releases of the interface artefacts are defined as milestones at the beginning of each development cycle and are used as a reference in the next development cycle.

4.3 Platform Engineering

The process group Platform Engineering aims at creating an infrastructure that implements (most of) the features needed to realize solutions based on adaptive services. Since the ASG platform consists of several subsystems, this process group includes both activities addressing the platform as a whole system and other activities addressing the individual subsystems within the platform. The activities *Analyse ASG Requirements*, *Design Overall Architecture*, *Select and Specify ASG Ontology Language*, *Model ASG Ontology*, *Platform Test* address the whole ASG platform. The activities *Analyse Subsystem Requirements*, *Design Subsystem*, *Implement Subsystem*, *Subsystem Test* are performed at least once for each individual subsystem.

Figure 10 shows an overview of all ASG Subsystems. The structure of the sub-system partially reflects the structure of the teams involved in developing the platform, since most of the subsystems is being developed by one team.

Figure 11 shows the activities included in the process group and the artefacts produced and consumed by the activities. Not every activity must be performed during one cycle: The diagram shows all relevant dependencies, i.e., all activities that can be potentially instantiated and all artefacts for which a new release can be produced. The diagram is intended to aid planning the development cycles, i.e., to decide which activities should be activated during a given development cycle to produce a new version of the related artefacts.
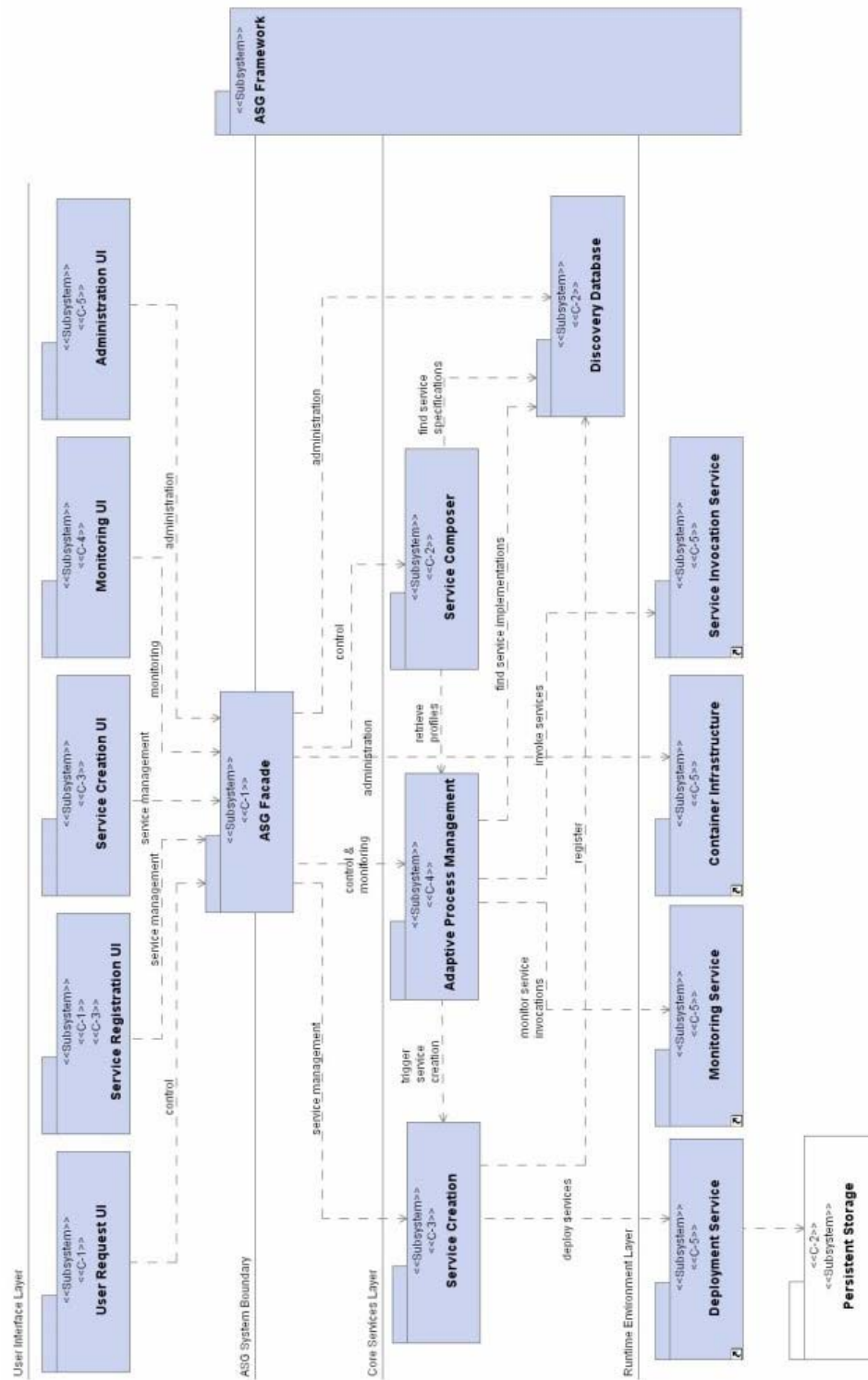
Figure 10:   Overview of ASG Subsystems

At the beginning, requirements for the ASG platform are elicited and analysed. The results of this activity (artefact ASG Requirements in the figure) are used to design the overall architecture (artefact ASG Architecture in the figure). With the architecture, all subsystems are identified. Thereafter, the requirements of each subsystem are analyzed and the results are collected in separate artefacts of the type Subsystem Requirements. The analysis of subsystem requirements is the first of the subsystem-related development activities that can be performed concurrently for each subsystem: Each subsystem development consists of a separate path of requirements analysis, design, implementation, and testing activities. At the same time, available formalisms for describing the ASG Ontology are investigated and the most suitable candidates are selected. Thereafter, the ASG Ontology is formalized using the selected language(s). The ASG Ontology is the language/data model used within the platform to exchange service-related information between subsystems, to specify user problems, expected solutions, and the solutions retrieved by the platform. Once matching versions of the subsystems and the ontology are available at the end of a development cycle, a new release of the platform can be integrated and tested (artefact ASG Platform Release in the figure).

The following artefacts are the results of the activities performed to engineer the platform: ASG Requirements, ASG Architecture, Subsystem Requirements, ASG Ontology Language, ASG Ontology, Subsystem Design, Subsystem, Tested Subsystem, ASG Platform Release.

The *ASG Requirements* list and describe the different features that the ASG Platform has to provide.

The *ASG Architecture* documents the architecture of the whole platform. Different architectural views such as logical, structural, functional, and behavioural views are provided.

For each subsystem, the requirements are identified and documented in an artefact *Subsystem Requirements*.

The artefact *ASG Ontology Language* describes the formalism to be used to formalize the ASG Ontology (i.e., Flora, WSMO, etc.).

The *ASG Ontology* artefact describes both the meta-model for specifying ASG services and the meta-model for formalizing the domain model.

The *Subsystem Design* documents the architecture of a single subsystem. For each subsystem, one artefact of this type is produced. Like the ASG Architecture, the Subsystem Design should also present different architectural views such as logical, structural, functional, and behavioural views.

The artefact *Subsystem* is an executable piece of code that implements the functionality of a single subsystem. For each subsystem, one artefact of this type is produced. The artefacts are, at the same time, configuration items managed with the ASG configuration management system.

The artefact *Tested Subsystem* includes the system tested, the test cases used for testing it, and the results of the test once it has been performed. For each subsystem, one artefact of this type is produced. The artefacts are at the same time configuration items managed with the ASG configuration management system.

The *ASG Platform Release* is an executable version of the ASG platform that represents a milestone and can be used as a reference for further development. The artefact is, at the same time, a configuration item managed with the ASG configuration management system.
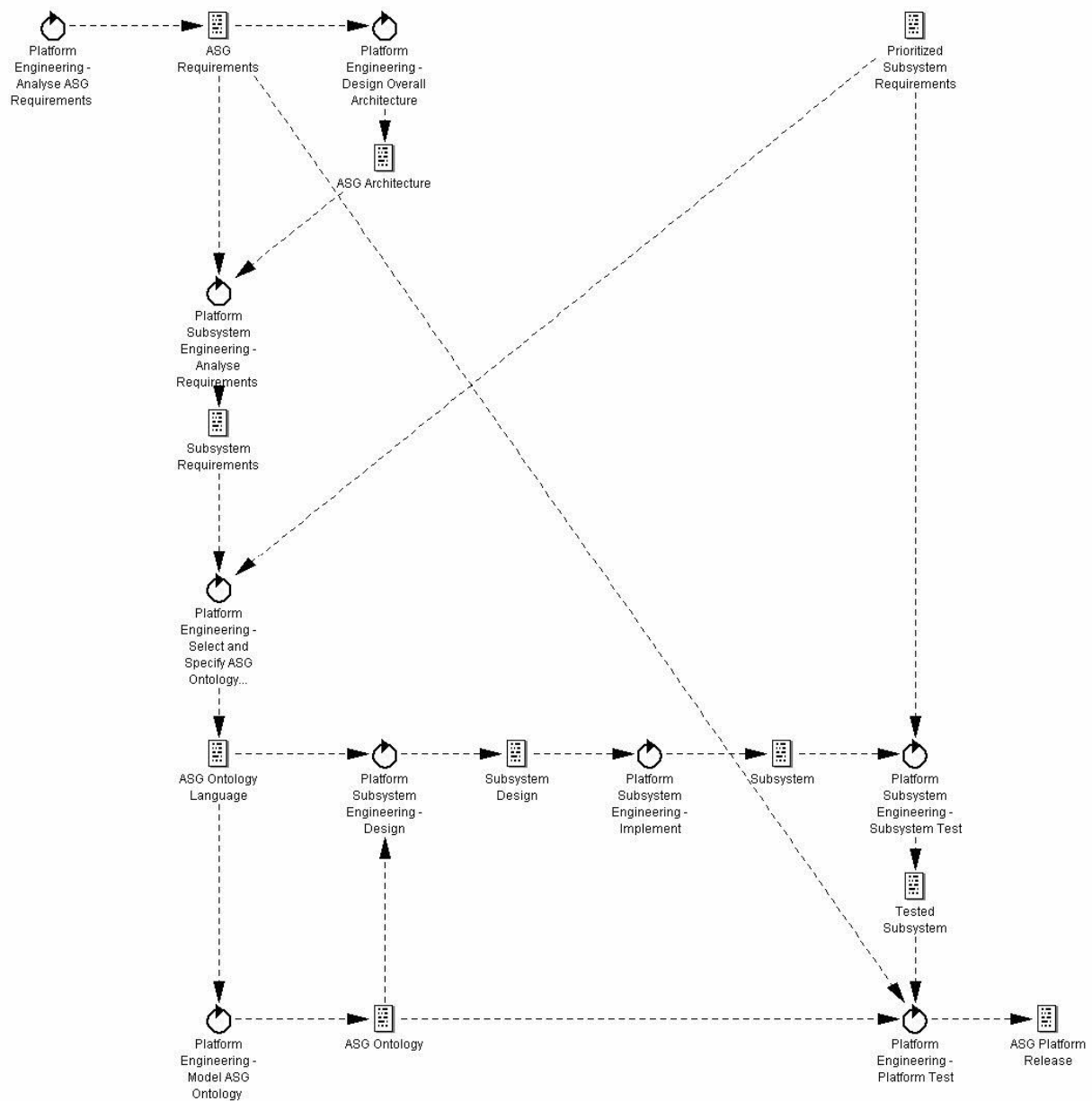
Figure 11:   Workflow with the Process Group Platform Engineering

As stated in the previous section, the artefact *Prioritized Subsystem Requirements* is used as input for the activities of the Platform Engineering process group but is produced by activities of the Application Engineering group. Thereafter, the Platform Engineering process is defined as an iterative, incremental process that is driven by the development of demonstrators.

The roles Customer[3], Requirements Manager, Architect, Ontology Manager, Subsystem Manager, Subsystem Requirements Engineer, and Platform Integrator are involved in the activities of the process group Platform Engineering.

In the following subsections, the individual activities are introduced in more detail.

4.3.1   Analyse ASG Requirements

The purpose of this activity is to establish the requirements for the ASG Platform.

The activity consists of the tasks:

- Identify the features of the ASG platform. All expected features of the platform are identified, documented, and agreed upon by the ASG project members.

- Derive correct and testable requirements. Single, unique, functional and non-functional requirements are derived. The consistency of the requirements is checked. The requirements are formulated in a way that their fulfilment after implementation can be objectively tested.

- Specify the Requirements by describing the possible interactions between applications and the ASG Platform. For this purpose, the use of use cases is recommended.

The activity is performed by the role Requirements Manager, Customer, Subsystem Manager. During the activity, the artefact ASG Requirements is produced. No explicit input artefact is consumed by the activity. The ASG Wiki server is used, among other things to exchange information about the ASG requirements and to publish related documents. The work component C4 has defined and used a template for requirements-related deliverables. Another asset for this activity are the Documentation Guidelines for Requirements.

- **Involved Roles**: Requirements Manager, Customer, Subsystem Manager

- **Tools**: ASG WIKI Server

- **Assets**: C4 Template for Requirements-related Deliverables, Documentation Guidelines for Requirements

- **Outputs**: ASG Requirements

---

[3] This role is challenging in every research project. In ASG, the role is mainly played by the industrial partners from the work component C7.

4.3.2 Design Overall Architecture

The purpose of the activity is to define the architecture for the ASG Platform by identifying all needed subsystems and their relationships according to the ASG requirements.

The activity consists of the tasks:

- Identify the different subsystems of the ASG Platform needed to implement the ASG requirements.
- Identify the relationships among the identified subsystems.
- Document the architecture according to the Documentation Guidelines. For this purpose, at least four different architectural views can be considered and should be documented: a logical, a structural, a functional, and a behavioural view.

  1. In the logical view, packages identify logical subsystems and dependencies identify interactions and data exchange among the logical subsystems.

  2. In the structural view, public interfaces of the subsystems shall be modelled in more detail.

  3. In the functional view, the functionality of each subsystem is described.

  4. In the behavioural view, the system behaviour is described for the case that a method is invoked.

- Baseline the Documentation of the ASG architecture.
- Define the Integration Strategy and Guidelines.
- Define the Integration Test Plan.

The activity is performed by the role Architect. During the activity, the artefact ASG Architecture is produced. The artefact ASG Requirements is consumed as input. The UML tool Magic Draw is used to model the system. The ASG Wiki server is used, among other things, to exchange information about the ASG Architecture and to publish related documents. The Documentation Guidelines for Analysis & Design can be used for further details on how to document the system.

- **Involved Roles**: Architect
- **Tools**: ASG WIKI Server, Magic Draw
- **Assets**: Documentation Guidelines for Analysis & Design, Magic Draw How-To
- **Inputs**: ASG Requirements

- **Outputs**: ASG Architecture

4.3.3 Analyse Subsystem Requirements

The purpose of this activity is to establish the requirements for each subsystem (identified during the activity Design Overall Architecture) by refinement of the ASG requirements (specified during the activity Analyse ASG Requirements) and the interactions between subsystems (identified during the activity Design Overall Architecture).

The activity consists of the tasks:

- Identify the interaction among this subsystem and the other subsystems as described in the ASG Platform Architecture.
- Define use case diagrams or sequence diagrams to describe the interaction.
- Document the interaction according to the Documentation Guidelines for Requirements.
- Document the identified requirements in the corresponding deliverable.
- Plan subsystem releases.

The activity is performed by the role Subsystem Requirements Engineer. During the activity, the artefact Subsystem Requirements is produced. The artefacts ASG Requirements and ASG Architecture are consumed as input. The tool Magic Draw is used to perform the activity. The Documentation Guidelines for Requirements can be used for further details on how to document the subsystem requirements.

- **Involved Roles**: Subsystem Requirements Engineer
- **Tools**: Magic Draw
- **Assets**: Documentation Guidelines for Requirements
- **Inputs**: ASG Requirements, ASG Architecture
- **Outputs**: Subsystem Requirements

4.3.4 Select and Specify ASG Ontology Language

The purpose of this activity is to identify and evaluate available formalisms for describing the ASG Ontology. The most suitable candidates are selected.

The activity consists of the tasks:

- Identify concepts and their relations used to specify the ASG services semantically and syntactically.

- Document the identified concepts and relations.

- Identify concepts and their relations used to specify the ASG domain ontology.

- Check both models in the common configuration management system.

- Identify available formalisms for modelling the concepts and relationships identified.

- Evaluate the formalisms with respect to their applicability within the ASG platform.

- Select the most suitable formalism(s).

The activity is performed by the role Ontology Manager. During the activity, the artefact ASG Ontology Language is produced, which consists of the selected formalisms and the rationales behind the decision made. The artefacts Prioritized Subsystem Requirements and Subsystem Requirements are consumed as input. The artefact Prioritized Subsystem Requirements is not produced within this process group but during activities from the Application Engineering group. The initial models of the ASG ontology needed to clarify requirements for the ontology language have been sketched with the tool Magic Draw. The tool Subversion is used as configuration management system. Furthermore, the tool ASG Wiki Server is used to exchange information related to the ASG Ontology Language and to publish the main results of this activity. Currently, no specific assets are used to perform the activity.

- **Involved Roles**: Ontology Manager
- **Tools**: Magic Draw, Subversion, ASG WIKI Server
- **Assets**: None
- **Inputs**: Prioritized Subsystem Requirements, Subsystem Requirements
- **Outputs**: ASG Ontology Language

4.3.5 Model ASG Ontology

The purpose of this activity is to model both concepts and relationships needed to specify services on the one hand, and the domain ontology on the other hand, by means of the selected ontology language.

The activity consists of the tasks:

- Model the concepts needed to specify the services and their relationships by means of the selected ontology language.

- Model the concepts needed to specify the domain ontology and their relationships by means of the selected ontology language.

- Check both specifications in the common configuration management system.

The activity is performed by the role Ontology Manager. During the activity, the artefact ASG Ontology is produced. The artefact ASG Ontology Language is consumed as input. The tool Magic Draw is used to perform the activity. Furthermore, the tool ASG Wiki Server is used to exchange information related to the ASG Ontology and to publish the main results of this activity. Currently, no specific assets are used to perform the activity.

- **Involved Roles**: Ontology Manager

- **Tools**: Magic Draw, ASG WIKI Server, Subversion

- **Assets**: None

- **Inputs**: ASG Ontology Language

- **Outputs**: ASG Ontology

4.3.6 Design Subsystem

The purpose of this activity is to define the design for the subsystem by identifying the different components and their relationships that are needed to implement a Subsystem that fulfils the respective Subsystem Requirements.

The activity consists of the tasks:

- Identify the different components needed to implement the Subsystem Requirements.

- Identify the relationships among the different components.

- Document the subsystem design with different (architectural) views as described in the Documentation Guidelines for Analysis & Design:

  – In the logical view, packages identify logical subsystems and dependencies identify interactions and data exchange among the logical subsystems.

  – In the structural view, public interfaces of the subsystems shall be modelled in more detail.

  – In the functional view, the functionality of each subsystem is described.

– In the behavioural view, the system behaviour is described for the case that a method is invoked.

- Define Subsystem Test Plan.

- Define Subsystem Integration Plan.

The activity is performed by the role Subsystem Manager. During the activity, the artefact Subsystem Design is produced. The artefacts ASG Ontology and ASG Ontology Language are consumed as input. Currently, no specific tool must be used to perform the activity. However, the UML tool Magic Draw is recommended; some teams also apply similar tools from other vendors. The asset Documentation Guidelines for Analysis & Design is used to perform the activity. The work component C4 defines and uses its own templates as a basis for the deliverables that collect the results of this activity (i.e., C4 Template for Design-related Deliverables).

- **Involved Roles**: Subsystem Manager

- **Tools**: Magic Draw

- **Assets**: C4 Template for Design-related Deliverables, Documentation Guidelines for Analysis & Design

- **Inputs**: ASG Ontology, ASG Ontology Language

- **Outputs**: Subsystem Design

4.3.7   Implement Subsystem

The purpose of this activity is to produce executable components/subsystems and verify that they properly reflect the subsystem design.

The activity consists of the tasks:

- Develop subsystem components by using the Java Coding guidelines.

- Develop Unit Tests for each component (executable component) for validating the internal behaviour of the whole subsystem (using, for example, JUnit).

- Execute each single Unit Test and thus validate the internal behaviour of the whole subsystem.

- If all Test Cases are successful, provide code files (configuration items) for the configuration management (Subversion) considering the ASG versioning concept.

- Develop Subsystem Test Cases for validating the component interaction using JUnit.

- Develop Subsystem Test Cases for validating the Subsystem Interaction using JUnit.

The activity is performed by the subsystem developers. However, the role Subsystem Manager is responsible for the activity and this is the only role visible for the project partners outside the development team. During the activity, the artefact Subsystem is produced. The artefact Subsystem Design is consumed as input. The tools Eclipse and JUnit are used to perform the activity. The asset Coding Guidelines is provided to improve the uniformity of code. The asset Unit Test Coverage Recommendations helps to identify the minimal coverage required for unit test.

- **Involved Roles**: Subsystem Manager
- **Tools**: Eclipse, JUnit
- **Assets**: Coding Guidelines, Unit Test Coverage Recommendations
- **Inputs**: Subsystem Design
- **Outputs**: Subsystem

4.3.8   Subsystem Test

The purpose of this activity is to perform an end-to-end testing of a Subsystem to ensure that it meets the Subsystem Requirements.

The activity consists of the tasks:

- Develop the Subsystem Test Plan.
- Perform the Subsystem Test.
- Discuss, document, and decide on the result of Test Cases (based on Subsystem Requirements and Design).
- Define a strategy for the regression test.
- Carry out regression testing in case of changes.
- Publish test results on the Maven site.
- Release the Subsystem by adding it to the ASG subversion server.

The activity is performed by the subsystem developers. However, the role Subsystem Manager is responsible for the activity and this is the only role visible for the project partners outside the development team. During the activity, the artefact Tested Subsystem is produced. The artefacts Subsystem Requirements, Prioritized Subsystem Requirements, and Subsystem are consumed as input.

The tool JUnit and Maven are used to perform the activity. Currently, no asset is used to perform the activity.

- **Involved Roles**: Subsystem Manager

- **Tools**: JUnit, ASG Subversion Server, Maven

- **Assets**: None

- **Inputs**: Subsystem Requirements, Prioritized Subsystem Requirements, Subsystem

- **Outputs**: Tested Subsystem

4.3.9 Platform Test

The purpose of this activity is to integrate the different subsystems, build up the ASG platform, and ensure that the different subsystems interact correctly according to the architecture.

The activity consists of the tasks:

- Ensure that all needed subsystems are present and tested (each subsystem is checked out from subversion in the correct version).

- Develop the Platform Test.

- Assemble a working build of the ASG platform.

- Perform the Platform Test.

- Document, discuss, and decide on the results of the Platform Test.

- Define a strategy for the regression test.

- Carry out regression testing in case of changes.

- Publish build and test results on the Maven site.

- Provide a working release of the ASG platform.

The activity is performed by the Platform Integrator with the collaboration of the Subsystem Managers. During the activity, a new ASG Platform Release is produced. The artefacts ASG Ontology, ASG Requirements, and Tested Subsystem are consumed as input. The tools Maven, Subversion, and JUnit are used to perform the activity. Currently, the assets Integration Guidelines, Integration Test Plan Recommendations, and Maven Guidelines are used to perform the activity.

- **Involved Roles**: Platform Integrator, Subsystem Manager

- **Tools**: Maven, Subversion, and JUnit

- **Assets**: Integration Guidelines, Integration Test Plan Recommendations, Maven Guidelines
- **Inputs**: ASG Ontology, ASG Requirements, Tested Subsystem
- **Outputs**: ASG Platform Release

5   Discussion

The discussion in this section focuses on process generalizability, process peculiarities, and open process issues. Most of the considerations are presented from the point of view of the process group aimed at engineering the ASG platform but also pertain to the whole development process and concern, in particular, the relationships between the process groups.

The generalizability of the platform engineering, i.e., the applicability of the process to other, similar contexts, is ensured by the fact that the process was defined on the basis of the software engineering processes described in the international standard ISO/IEC 12207 "Software Lifecycle Processes" and includes typical activities such as requirements analysis, design, implementation, integration, and testing, which are performed in the majority of the projects aimed at developing software. Also, the process described does not present significant differences from other development processes in terms of activity performed. This is mainly due to the fact that the process aims at developing the ASG platform, whereas peculiar activities are required to use it. ASG-specific activities characterise, therefore, more the process groups Application and Service Engineering (described in [32]).

However, the processes discussed present own peculiarities. One first interesting characteristic is how the different process groups relate to each other: an ASG solution is based on at least one application operated by an end user (usually a human being) and on services that are discovered, negotiated, composed, invoked, and, eventually, replanned by the platform. This means that a single solution consists of several parts such as an application, services, and the ASG platform, which are engineered by different teams or even organisations and need to be integrated and tested in different stages. A similar situation can also be found in a solution developed following a service-oriented approach without semantic-based interoperability support. In order to deal with such complexity, the ASG development process is defined as an iterative, incremental process that is driven by the development of demonstrators. On the other hand, since the ASG project is a research project, the demonstrators implemented do not only solve a customer's problems but must primarily show the functionality of the platform and the achievement of research goals. Therefore, the development process turns out to be somehow technology-driven. Sometimes the question of what kind of problems can be solved with the technologies under investigation seems to influence the development activities more that the question of what kind of technology one needs to solve a given, concrete problem. The various integration stages also imply several testing stages. The needed testing activities could already be identified in this phase of the project but

were not sufficiently investigated yet. In particular, goals and strategies of the different test activities must be analyzed deeper in order to avoid both redundant, time-consuming testing and insufficient test coverage.

Another characteristic, which is quite usual in research projects, is that enacting development activities in strict accordance with the process description can not and should not be enforced. This is a consequence of the fact that the inherent creativity of the research work must be defended and supported. The prototypes developed represent a suitable means for making ideas explicit and enhance the communication among teams. On the other hand, such prototypes aim at showing the achievements of research activities and the quality assurance activities performed within the project must focus more on the quality of these achievements than on the quality of the software developed.

Another peculiarity of the process is the great amount of emerging standards and languages to be considered. Standards from the field of service-oriented architectures such as Web services play an important role as do ontology languages such as OWL and WSML and semantic service approaches such as OWL-S and WSMO.

The process is also influenced by the great diversity of the development teams in terms of team size, programming skills, and domain of interest. A shared architecture modelled with the aid of tools that allow distributed design over the Internet is the key to aligning ideas and letting a common vision emerge from separate sets of, often only implicitly stated, requirements. The subsystems within the shared architecture partially reflect the structure of the development team, and development cycles are also planned in terms of parts of the architecture that should be addressed and implemented.

6   Summary and Outlook

This deliverable presents the first version of the process for developing the ASG platform. It also describes the methodology applied to formalize and validate the process.

The many activities performed to define and establish a shared development process and the great amount of effort spent on them are justified by the fact that development activities are performed within the project by several teams from different organisations. Development teams range from two-person teams consisting of a PhD student and a master student to ten professional programmers. Development teams are not collocated and team members speak different native languages. For all these reasons, there is a strong need in the project for a common taxonomy of the terms related to the software development process applied.

This deliverable contributes both a process architecture (see section 4.1) and a process description (sections 4.2 and 4.3). The work on software processes within the ASG project focuses on several aspects addressed from different points of view. This deliverable addresses, in particular, the activities aimed at engineering the ASG platform. A description of the processes intended for engineering services and applications for the platform can be found in deliverable D6.III-2 "ASG Development Process – Application and Service Engineering". An overview of all development process-related deliverables and their relationships is presented in section 1.3.

Describing processes explicitly is just one of the activities performed to manage the ASG development process. Other activities include establishing, analysing, and improving the process continuously. The process management approach followed in the project is also presented in this deliverable (section 3.2). According to the approach, this deliverable represents another step towards the challenging formalisation of development processes in the emerging domain of solutions based on semantic services. Further steps to be performed in the near future include a deeper analysis and implementation of the many testing activities needed to ensure reliable, integrated solutions. Furthermore, a post-mortem analysis of the development activities should be performed to gather and classify the lessons learned during two and a half years of development.

The process description presented in this deliverable is used as a reference for planning, performing, and monitoring the ASG development activities. Therefore, the deliverable targets all parties involved in the development of the plat-

form[4] and those people who wish or need more insight into the various interrelated development activities.

---

[4] ASG members involved in development activities may appreciate the electronic ASG development process guide available at
https://asg-platform.org/cgi-bin/twiki/viewauth/Internal/ASGDevelopmentProcess.

References

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

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

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

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

[5] Becker-Kornstaedt, Ulrike; Hamann, Dirk; Kempkens, Ralf; Rösch, Peter; Verlage, Martin; Webby, Richard; Zettel, Jörg: Support for the Process Engineer. The Spearmint Approach to Software Process Definition and Process Guidance. In Advanced Information Systems Engineering. International Conference CAiSE'99 - Proceedings (1999), 119-133.

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

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

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

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

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

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

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

[13] Liu, W.; Goldzmidt, G.; Joseph, J.: 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

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

[15] 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).

[16] Motta, E.; Dominguez, J.; Cabral., L.; Gaspari, M.: 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.

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

[18] 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.

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

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

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

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

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

[24] Software Process Engineering Metamodel Specification, January 2005, Version 1.1, formal/05-01-06, an adopted specification of OMG Group Inc. Available at http://www.omg.org/technology/documents/formal/spem.htm

[25] Vetere G., Lanzerini M.: Models for semantic interoperability in service-oriented architectures. IBM Systems Journal, Vol. 44, No. 4, 2005Zettel, 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)

[27] Zimmermann, O.; Krogdahl, P.; Gee, C.: 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/

ASG Deliverables

[28] Bella et al.: Adaptable Process Engineering Survey, ASG Deliverable D6.III.1, delivered at M6

[29] Eisenbarth et al.: Requirements Specification Survey, ASG Deliverable D6.I-1, delivered at M6

[30] Eisenbarth et al.: Reuse-Oriented Requirements Technique, ASG Deliverable D6.I-2, delivered at M12

[31] Eisenbarth et al.: Case Study: Requirements Specification, ASG Deliverable D6.Il-1, delivered at M18

[32] Lehner et al.: ASG Development Process – Application and Service Engineering, ASG Deliverable D6.III-2, delivered at M18

[33] Tahir: Testing Methodology for ASG Applications and Services, ASG Deliverable D6.III-5, delivered at M18

[34] Tahir: Tracing and Logging Concept, ASG Deliverable D6.IV-2, delivered at M18

[35] Tahir: Testing Methodology for Platform Code, ASG Deliverable D6.IV-3, delivered at M18

Project Consortium Information

Partner	Acronym	Contact
University of Potsdam, Germany		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	Fraunhofer Institut Experimentelles Software Engineering	Dr. Dirk Muthig Fraunhofer Institut Experimentelles Software Engineering. Fraunhofer Platz 1, D-67663 KAISERSLAUTERN Germany Email: muthig@iese.fraunhofer.de Tel: +49-631-6800-1320
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 Anwendungszentrum fuer Software-, Informations- 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	DI Klaus Jank Siemens AG Corporate Technology, Software and System Architecture Otto-Hahn-Ring 6 D-81730 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

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
University of Koblenz-Landau, Germany		Prof. Steffen Staab Institute of Computer Science Universität Koblenz-Landau PO Box 201 602 56016 Koblenz Germany Email: staab@uni-koblenz.de Tel: +49-261-287 2761
Erik Lillevold, Norway		Erik Lillevold Åsheimneien 33 2016 Frogner Norway Email: erlille@online.no Tel: +47-9134-4641

Document Information

Title: ASG Platform Development
Process
Deliverable D6.IV-1

Date: March 3, 2006
Report: IESE-137.06/E
Status: Final
Distribution: Public