

Project Number: **215219**
 Project Acronym: **SOA4All**
 Project Title: **Service Oriented Architectures for All**
 Instrument: **Integrated Project**
 Thematic Priority: **Information and Communication Technologies**

D7.3 End User Service Design

Activity:	Activity 3 - Use Case Activities	
Work Package:	WP 7 - End-user Integrated Enterprise Service Delivery Platform	
Due Date:	M13	
Submission Date:	17/04/2009	
Start Date of Project:	01/03/2006	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	SAP	
Revision:	1.12	
Authors:	Juergen Vogel SAP Florian Schnabel SAP Florian Stroh SAP Lai Xu SAP Patrick Un SAP Nikolay Mehandjiev UNIMAN Usman Wajid UNIMAN Freddy Lecue UNIMAN Tomás Pariente Lobo ATOS	
Reviewers:	Ivan Delchev SAP Reto Krummenacher UIBK	

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
1.1	30.11.2008	First Version	Juergen Vogel (SAP)
1.2	13.03.2009	Update Section 2	Juergen Vogel (SAP), Usman Wajid (UNIMAN)
1.3	17.03.2009	Update storyboard (Annex A), Section 4	Juergen Vogel (SAP), Florian Schnabel (SAP), Lai Xu (SAP)
1.4	20.03.2009	Update Section 4	Florian Stroh (SAP), Patrick Un (SAP)
1.5	20.03.2009	Contribution to Section 4.4	Freddy Lecue (UNIMAN)
1.6	25.03.2009	Contribution to Sections 3 and 4	Tomás Pariente Lobo (ATOS)
1.7	26.03.2009	Update Sections 4, 5, and 6; Input Section 7	Juergen Vogel (SAP), Florian Stroh (SAP), Florian Schnabel (SAP), Patrick Un (SAP), Lai Xu (SAP), Nikolay Mehandjiev (UNIMAN), Usman Wajid (UNIMAN), Freddy Lecue (UNIMAN)
1.8	27.03.2009	Move listings to Annex, Update all Sections, add Executive Summary, Sections 1 and 8	Juergen Vogel (SAP)
1.9	30.03.2009	Introduction, Conclusions, Update Section 4.8	Juergen Vogel (SAP), Lai Xu (SAP)
1.10	30.03.2009	Some minor corrections	Juergen Vogel (SAP), Patrick Un (SAP)
	06.04.2009	Internal review by Ivan Delchev (SAP)	
1.11	07.04.2009	Corrections according to review	Juergen Vogel (SAP)
	09.04.2009	Internal review by Reto Krummenacher (UIBK)	
1.12	15.04.2009	Corrections according to review	Juergen Vogel (SAP)
Final	20.04.2009	Overall format and quality revision	Malena Donato (ATOS)

Table of Contents

TABLE OF CONTENTS	3
LIST OF FIGURES	4
GLOSSARY OF ACRONYMS	5
EXECUTIVE SUMMARY	7
1. INTRODUCTION	8
1.1 PURPOSE AND STRUCTURE OF THE DOCUMENT	9
1.2 FUTURE WORK	9
2. SUMMARY OF REQUIREMENTS	10
3. ARCHITECTURE	13
4. FUNCTIONAL AND TECHNICAL SPECIFICATION OF COMPONENTS	16
4.1 FRONT END (GUI)	16
4.2 USER MANAGEMENT	17
4.3 SERVICE AND PROCESS SELECTION	18
4.4 ANNOTATIONS AND COMMUNITY SUPPORT	19
4.5 PROCESS MODELING	21
4.6 PROCESS STORAGE	22
4.7 PROCESS DEPLOYMENT AND EXECUTION	22
4.8 PROCESS MONITORING	23
5. SELECTED SERVICES FOR THE DEMONSTRATOR	25
5.1 PUBLIC SERVICES	25
5.2 SERVICES FOR HANDLING HUMAN TASKS	26
5.3 SAP ENTERPRISE SERVICES	27
5.3.1 <i>General SAP Architecture, Data Models and Access</i>	27
5.3.2 <i>ES Bundle: Records and Document Management</i>	30
6. SEMANTIC ADAPTATION AND INTEGRATION LAYER FOR SAP ES	33
6.1 CLASSIC WEB SERVICE ARCHITECTURE	34
6.2 SEMANTICALLY-ANNOTATED WEB SERVICES	35
6.3 AN ARCHITECTURE FOR SEMANTICALLY-ANNOTATED ES	37
7. EVALUATION WORKSHOP	39
7.1 DESIGNS FOR COMPOSING WEB SERVICES	40
7.2 DESIGN RATIONALE	40
7.3 TOP-LEVEL ISSUES	40
8. CONCLUSIONS	43
9. REFERENCES	44
ANNEX A: UPDATED SCENARIO “REGISTRATION OF A BUSINESS”	46
ANNEX B: EXAMPLES FOR SERVICE ANNOTATIONS	81

List of Figures

Figure 1: SOA4All Overall Architecture.....	13
Figure 2: Overview of Adaptations and Extensions to SOA4All	15
Figure 4a: Extended Attribute List for Process Annotations	20
Figure 4b: Extended Element List.....	20
Figure 5: Process Deployment and Execution	23
Figure 6: Invocation of the Web Service ValidateCreditCard within the process “Registration of a Business” V2	25
Figure 7: Invocation of the Web Service ValidateUKAddress within the process “Registration of a Business” V2	26
Figure 8: Overall SAP NetWeaver Architecture with Processes, ES, and Enterprise Systems (in the style of [König2004])	29
Figure 9: WS Navigator for Enterprise Service “Create Public Sector Document”.....	30
<i>Figure 10: Classic Web Service Architecture.....</i>	<i>34</i>
Figure 11: Web Service Architecture with Semantic Layer	36
<i>Figure 12: Web Service Architecture with Semantic Layer & Integration Layer.....</i>	<i>38</i>
Figure 13: gIBIS Design Rationale for Composition (Issue 1)	42
Figure 14: gIBIS Design Rationale for Composition (Issue 2)	42

Glossary of Acronyms

Acronym	Definition
API	Application Programming Interface
BPEL	Business Process Execution Language
BPM	Business Process Modeling
BPMN	Business Process Modeling Notation
CMS	Content Management System
CRM	Customer-Relationship Management
D	Deliverable
DSB	Distributed Service Bus
EC	European Commission
EJB	Enterprise Java Beans
EP	Enterprise Portal
ERP	Enterprise Resource Planning
ES	Enterprise Service
ESR	Enterprise Service Repository
ESB	Enterprise Service Bus
EU	European Union
EUD	End User Development
GDT	Global Data Type
GUI	Graphical User Interface
HCM	Human Capital Management
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifier
ISO	International Organization for Standardization
IT	Information Technology
M	Median
NLP	Natural Language Processing
OWL	Web Ontology Language
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	RDF Schema

REST	REpresentational State Transfer
SaaS	Software as a Service
SAWSDL	Semantic Annotations for WSDL
SCM	Supply Chain Management
SD	Standard Deviation
SEE	Semantic Execution Environment
SEI	Service Endpoint Interface
SME	Small and Medium Enterprise
SOA	Service-Oriented Architecture
SOA4All	Service-Oriented Architectures for All
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SRM	Supplier Relationship Management
TCO	Total Costs of Ownership
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USD	United States of America Dollars
WP	Work Package
WS	Web Service
WSDL	Web Services Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
XI	Exchange Infrastructure
XML	Extensible Markup Language
XQuery	XML Query Language
XSPARQL	XQuery for SPARQL

Executive Summary

WP7 is one of the three SOA4All use cases and has the public sector as its target domain. It envisions an integrated service delivery platform that will be realized based on the technologies and tools developed in SOA4All. This platform will allow non-technical end users in public administrations to handle typical administrative procedures (such as a permit approval process). More specifically, using the Web-based tools of the SOA4All Studio, public servants of various governmental organizations can search, model, annotate, modify, share, analyze, and execute administrative procedures in the form of lightweight business processes. These processes may be composed of SAP Enterprise Services, public Web services (hosted by 3rd party service providers), and human activities (to be executed by end users). Thus, the main result of WP7 will be an integrated demonstrator that addresses the specific needs of public administrations. For public administrations, the main benefit of such a flexible, open, and shared service delivery platform is the possibility to quickly address new challenges and requirements, e.g., such as the ones formulated by the EU Services Directive. Citizens and businesses that interact with public administrations over the platform will benefit from faster and simpler public services. This deliverable serves as a guideline and as a technical specification for implementing the service delivery platform based on the technical components as well as the Web-based front end developed within the SOA4All project. The deliverable also includes the design of a semantic adaptation and integration layer for handling SAP Enterprise Services, a detailed and updated version of the use case story board that demonstrates the features from the user's point of view, and a description of the first evaluation workshop for validating the project's ideas.

1. Introduction

WP7 is one of the three SOA4All use cases and has the public sector as its target domain. Public administrations nowadays have to deal with hundreds of different procedures (e.g., for handling a parking permit request) that are typically implemented in one or more legacy systems or even executed manually. At the same time, the increasing number of regulatory changes and new regulations, including an increasing number of international, bilateral agreements, asks public administrations to constantly adapt their procedures in a flexible and cost-efficient way. For instance, the EU Services Directive requires administrations to implement a one stop e-Government approach where constituents can file requests for public services via a single point of contact. This single point of contact then coordinates all necessary activities, which is contrary to the current setup where the constituents themselves have the main responsibility and need to manage on their own. As a consequence, public administrations now need to adapt their service offerings to the needs of each constituent.

SOA4All investigates different key technologies (Semantic Web services, context adaptation, Web 2.0 principles) that can help to address such challenges on the basis of an advanced service-oriented architecture. WP7 envisions an open and flexible service delivery platform where administrative procedures are handled over a central Internet portal as single point of contact. Administrative procedures are composed of Semantic Web services. These services can be combined in different ways so that new procedures can be created or existing ones can be adapted easily. A key element for creating the content for this service delivery platform in an efficient and scalable way is the enablement of end (or business) users that resemble the large majority of employees in public administrations (and other organizations). Such business users have a detailed understanding of the procedures in their field of expertise but lack the specific programming skills required to actively consume and compose Web services. The SOA4All approach therefore is to provide end users with simple Web-based tools on top of Semantic Web services so that they can search, model, annotate, modify, share, analyze, and execute administrative procedures on the basis of Web services. This Web 2.0 approach is also a main differentiator where SOA4All advances the current state of the art.

The main goal of WP7 is to realize this service delivery platform as an integrated demonstrator. This demonstrator will be built from the components developed by the technical WPs of the project: the communication and data storage infrastructure will be provided by WP1, formalisms and tools for the semantic handling of services by WP3, services discovery and a service registry by WP5, lightweight process modeling and execution by WP6, and different Web-based end user tools by WP2. Besides the technical integration and validation, the main contribution of WP7 from a technological point of view will be to investigate how so-called Enterprise Services can be integrated into the open, dynamic, lightweight, and end user-driven service platform that is envisioned by SOA4All. Such Enterprise Services, which are provided by SAP in the scope of WP7, offer complex business functionality like the management of resources or relationships with customers. But at the same time, they typically have large syntactic (i.e., WSDL-based) service interface descriptions that are difficult to understand for non-expert service consumers. Thus, by investigating how to make Enterprise Services consumable for non-experts, WP7 will significantly increase the number of services that can be handled by SOA4All.

Previously, D7.2 motivated the public sector as a target domain for the SOA4All project and described how the research of SOA4All can support SAP in realizing a novel service delivery platform for the public sector by means of an exemplary use case scenario. The resulting technical and business requirements for this service delivery platform were described in D7.1. The contribution of this deliverable is to illustrate how the SOA4All components will be leveraged to build the service delivery platform. In addition, WP7 will extend and customize

the graphical front end, develop dedicated services for the handling of human tasks, and provide a semantic adaptation and integration layer for using SAP Enterprise Services within SOA4All.

1.1 Purpose and Structure of the Document

The purpose of this deliverable is to describe the use, adaptation, and extension of the components provided by technical WPs WP1 – WP6, which are necessary to implement the service delivery platform for the public sector in a way to meet the requirements detailed in D7.1 and D7.2. In Chapter 2, these requirements are shortly summarized. Chapter 3 gives an overview of the SOA4All architecture with front end and back end components and also highlights which components will be customized for the use case demonstrator. Chapter 4 details these modifications and extensions. Chapter 5 describes the business services that will be used for the demonstrator with a focus on Enterprise Services that SAP offers for its customers from the public sector. Annex A presents an updated version of the use case storyboard (presented first in D7.2) and as such demonstrates the functionalities and benefits of the service delivery platform from the user's perspective. The integration of the SAP Enterprise Services into the SOA4All architecture by means of a semantic adaptation and integration layer is discussed in Chapter 6. Examples for semantic service annotations can be found in Annex B. Chapter 7 elaborates on the first evaluation workshop that will be conducted in the upcoming months. Finally, Chapter 8 concludes this deliverable.

1.2 Future Work

Following the technical design of the envisioned service delivery platform in this deliverable, a first prototype will be implemented until August 2009 (M18). The status of that demonstrator will be documented in D7.4. The semantic adaptation and integration layer for handling SAP Enterprise Services including the necessary semantic descriptions of the services to be involved in the use case scenario will be available with D7.6 in February 2010 (M24). The final version of the demonstrator will be delivered with D7.5 in (M33). The focus in the last three months of the project is on the concluding technical validation and the user experience evaluation of the achieved results. The corresponding report D7.7 will be available in February 2011 (M36).

2. Summary of Requirements

In this chapter, we summarize and update the most important functional and non-functional requirements for the envisioned service delivery platform that has previously been described in Deliverables D0.2, D7.1, and D7.2. These requirements are the basis for the design of the demonstrator. They were derived from the general environment and intended use of the service delivery platform as well as the profile of end users in the public sector (e.g. city council workers and administrators).

SOA4All aims to empower ordinary end users to combine and consume large numbers of services. Service mashups have been identified as central elements that can address IT challenges in public sector [Gartner2007]. However, a general overview of public sector administrators illustrates that they are more focused on the managerial aspects of their work rather than technical aspects [Swain1995]. This behavior can be attributed to the lack of technical knowledge of public sector workers.

As in our case, it is clear that the SOA4All target end users in the public sector will not have a software development background. Their main interest would be to use SOA4All to fulfill their routine job tasks in a flexible and context specific manner. In this respect, [Swain1995] notes that the availability of user-friendly software packages seems to increase end-user computing and public sector administrators appears to encourage it. These results promote graphical representation of software solutions like SOA4All for users with average IT skills so that a majority of public sector administrators / workers can effectively handle business processes and compose mashups from existing services.

An important aspect of understanding the end users is to carry out preliminary studies to ensure all end user requirements are collected. A personal approach to gather such requirements may involve interviews, brainstorming sessions, and use cases and storyboards. These techniques can be incorporated within workshops where all project stakeholders are gathered together to discuss the project needs and goals. In this respect, a set of workshops has been planned to gather further requirements about end users from the public sector (see Chapter 7).

The following table summarizes the most important requirements, some of them being specific for WP7 while others apply to all SOA4All use cases:

Requirement	Description
Shared Service Registry and Process Repository	The service delivery platform should provide a central and shared registry that lists all services and processes as well as a shared repository for processes, including those that are added by the target users so that existing processes can be executed, re-used, and modified by users other than the original author.
Support SAP Enterprise Services	The service delivery platform should allow accessing SAP Enterprise Services that provide rich business functionality for administrative procedures such as the management of the documents associated with a certain case. Enterprise Services are typically hosted by the internal IT department of a public administration, but could also be hosted by centralized shared services centers for several administrations or independent 3 rd providers. Currently, SAP Enterprise Services have complex, WSDL-based service interfaces that can be handled by service experts only.

Requirement	Description
	The integration of SAP Enterprise Service into the SOA4All platform therefore requires (1) a semantic adaptation and integration layer that is compatible with the semantic formalisms used in SOA4All and (2) a front end that makes the handling of such services feasible for the target user group.
Support Public Web Services	The service delivery platform should support the integration of Web services that are offered by external public or private service providers so that a public administration can outsource certain functionality.
Support Human Tasks	Business processes in public administrations often involve activities that are executed by humans such as checking requests for completeness and correctness. The envisioned service delivery platform therefore needs to support the modeling and execution of human tasks .
Manage and Search Services and Processes	In a typical public administration, the service delivery platform will contain several hundred services and processes so that intelligent search and organization techniques are required to minimize the access times for users.
Execute Services and Processes	The service delivery platform should allow the seamless execution (i.e., consumption) of services and processes.
Capture User Knowledge on Services and Processes	<p>The main efficiency gain from social Web 2.0 applications results from capturing and structuring knowledge from all users in an organized way and from making this knowledge easily accessible to all users. In addition to managing the services and processes in a shared registry, the service delivery platform should allow to manage different types of meta-data, which supports the handling of services and processes for the SOA4All tools or the SOA4All users:</p> <ul style="list-style-type: none"> • Semantic annotations for services and processes to support automatic service discovery and service composition • Categories to organize services and processes • Tags to organize services and processes • Ratings as user feedback on the quality of services and processes • Textual Comments for user discussions, guidelines, or qualitative feedback
Compose New Processes	Users of the service delivery platform should be able to compose new processes that resemble administrative procedures by connecting activities with control flow elements (i.e., business logic). Activities may be SAP Enterprise Services, public Web services, or human tasks.

Requirement	Description
Modify Existing Processes	The modification of existing processes is needed so that they can be reused as parts for new ones or in order to adapt them to new requirements.
Analyze and Access Status Information on Services and Processes	The service delivery platform should provide status information on selected process instances so that users can identify the active activity of a running instance or can retrieve information about the success or failure of a completed or aborted instance. Moreover, users can evaluate the performance of a service before placing it in a process model, which is of particular interest for public Web services from 3 rd party providers.
Support User Roles	Like in other professional organizations, employees in public administrations have different positions with specific rights and responsibilities. The service delivery platform should reflect this internal organization of administrations by supporting a user and role model that allows assigning specific access rights to data or functionalities to certain users or roles.
Usability	The main target user group of the service delivery platform will be business users, i.e., civil servants in public administrations with sufficient business knowledge to handle their assigned working tasks, but with limited IT knowledge (usage of office, Internet, and business applications but no programming or SOA skills). The user experience of the tools available via the service delivery platform should meet the specific needs of this target group by using a minimalist screen design with direct manipulation of all elements, by using human-computer interaction paradigms known from other applications like an Internet browser, by hiding the underlying technical complexity, by pre-filtering and sorting large amounts of information , by error tolerance, by offering explicit support and guidance for specific tasks, etc.
Low Administrative Overhead	A key business goal of the service delivery platform is to achieve a smaller total cost of ownership (TCO) when compared to traditional software solutions. Besides enabling business users with end user development (EUD) facilities and thus shifting efforts from IT specialists to end users, this requirement can be accomplished by low installation and maintenance costs for all software components of the platform.

3. Architecture

The architecture of WP7 is based on the SOA4All architecture as described in D1.4.1A. The overall architecture of SOA4All can be divided into four parts: SOA4All Studio, Distributed Service Bus, SOA4All Platform Services, and Business Services (3rd party Web services and lightweight processes). Each of these parts is subject to dedicated work packages of the project. This deliverable presents the project's overall approach towards a SOA4All service delivery platform and defines how the various parts integrate and interact. Figure 1 shows a high-level architecture of the SOA4All platform.

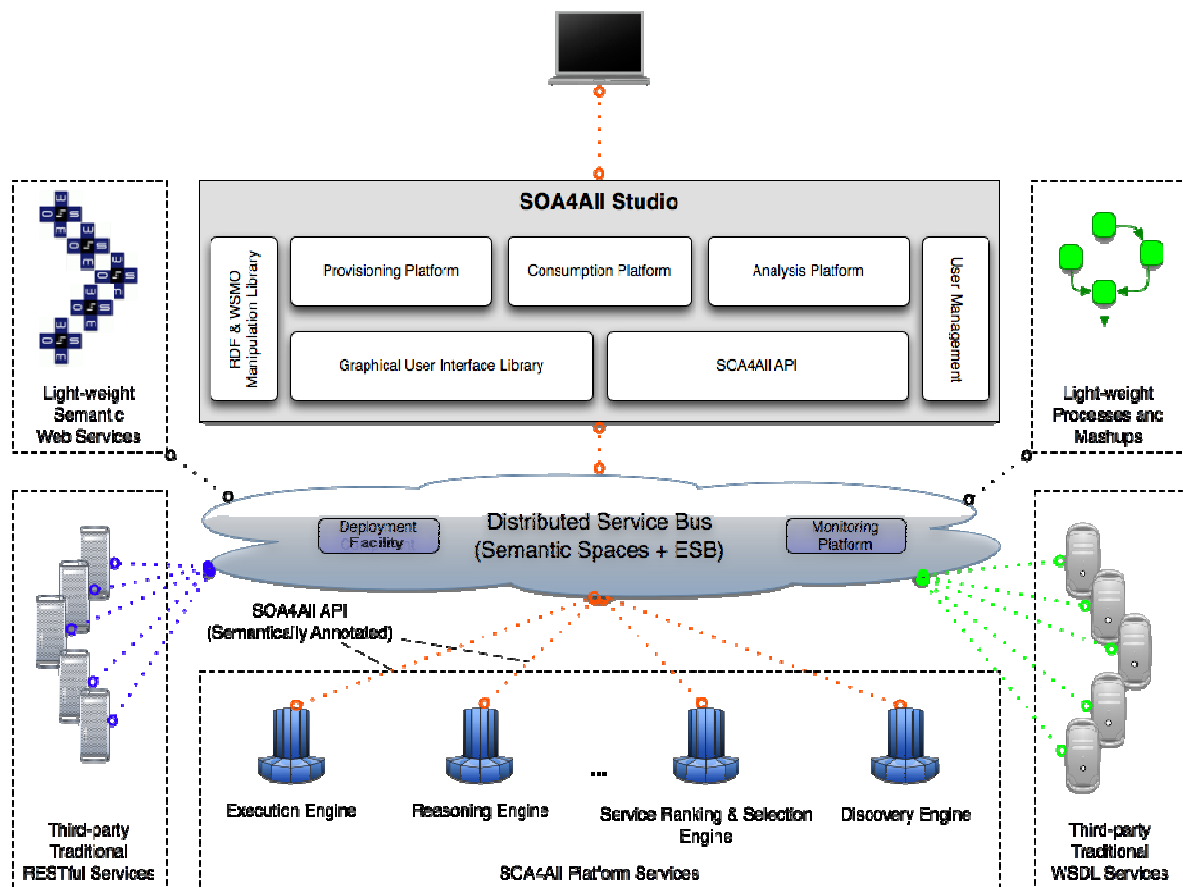


Figure 1: SOA4All Overall Architecture

The SOA4All overall architecture is composed of four main building blocks:

- Distributed Service Bus:** The Distributed Service Bus (DSB) is developed in WP1 and serves as infrastructure service and core integration platform and is a direct evolution of the open-source PETALS Enterprise Service Bus (ESB). Furthermore, the DSB is extended by a scalable Semantic Space infrastructure, used as a shared memory to build repositories, as cooperative access to monitoring data, and as communication infrastructure to enhance the traditionally message-oriented bus towards a publication infrastructure for anonymous and asynchronous service communication with a notion of event-driven architectures.
- SOA4All Studio:** This is the front end of SOA4All. The SOA4All Studio is developed in WP2 and delivers a fully Web-based user front end that enables the creation,

provisioning, consumption, and analysis of the platform services and various 3rd party business services that are published to SOA4All. The studio supports different types of users (service composers, service annotators, and service consumers as defined in Section 5.3, D7.2) at different times of interaction.

- **Platform Services:** These services are products of WP3, WP5, and WP6 and deliver service discovery, ranking and selection, composition and invocation functionality, respectively. These components are exposed to the SOA4All Distributed Service Bus as Web services and are hence consumable as any other published service. Their functionalities are used by the SOA4All Studio to offer clients the best possible functionality, while their combined activities (i.e., discovery, selection, composition, and invocation) are coordinated via the DSB. The ensemble of DSB, SOA4All Studio, and platform services delivers the innovative, fully Web-based and Web-enabled service experience of the SOA4All project: global service delivery at the level of the bus, Web-style service access via the Studio, and advanced state-of-the-art service processing, management, and maintenance via dedicated platform services.
- **Business Services (Web Services) and Processes:** These are 3rd party business services that are annotated semantically and are composed by means of the SOA4All infrastructure. These invocable services can be defined as RESTful or WSDL-based services. The semantic descriptions are published in the Service Registry in the formats of MicroWSMO or WSMO-Lite (as specified in WP3), respectively, and are used for reasoning with service capabilities (functionality), interfaces, and non-functional properties, as well as with context data. These semantic descriptions are the main enablers of the automation processes related to Semantic Web services. The services can be composed and executed using lightweight processes and mashups. Both, mashups and semantic descriptions of service compositions are published to the shared Semantic Spaces, and become a public good for automated large-scale service computing.

The envisioned service delivery platform makes use of all of these components. However, in some cases use case-specific adaptations or extensions are required to comply with the requirements stated in Chapter 2. Figure 2 shows a high-level overview of the additions that should be provided to adapt the overall architecture to the specific needs of WP7:

1. SOA4All Studio
 - a. Provisioning Platform: In case SAP Enterprise Services should be annotated semantically using the SOA4All Provisioning tools these need to take into account the peculiarities of these services such as their large and complex service interfaces (WSDLs) and their logical dependencies (pre and post conditions) when comparing them to “regular Web services”.
 - b. Consumption Platform: The specific needs of the public administration’s processes will be developed in this module (process templates, wizards for the SOA4All Composer, etc.).
 - c. User Management: The SOA4All Studio provides the basic user interface and services to allow for user management but currently does not provide an elaborate role model as required for the scenario.
2. For integrating state-of-the-art SAP Enterprise Services into the SOA4All platform, a Semantic Adaptation and Integration Layer will be developed.
3. Support for human tasks: Human tasks involve the usage of a specific user interface that shows to the end user waiting tasks (if any) and a user interface for the actual

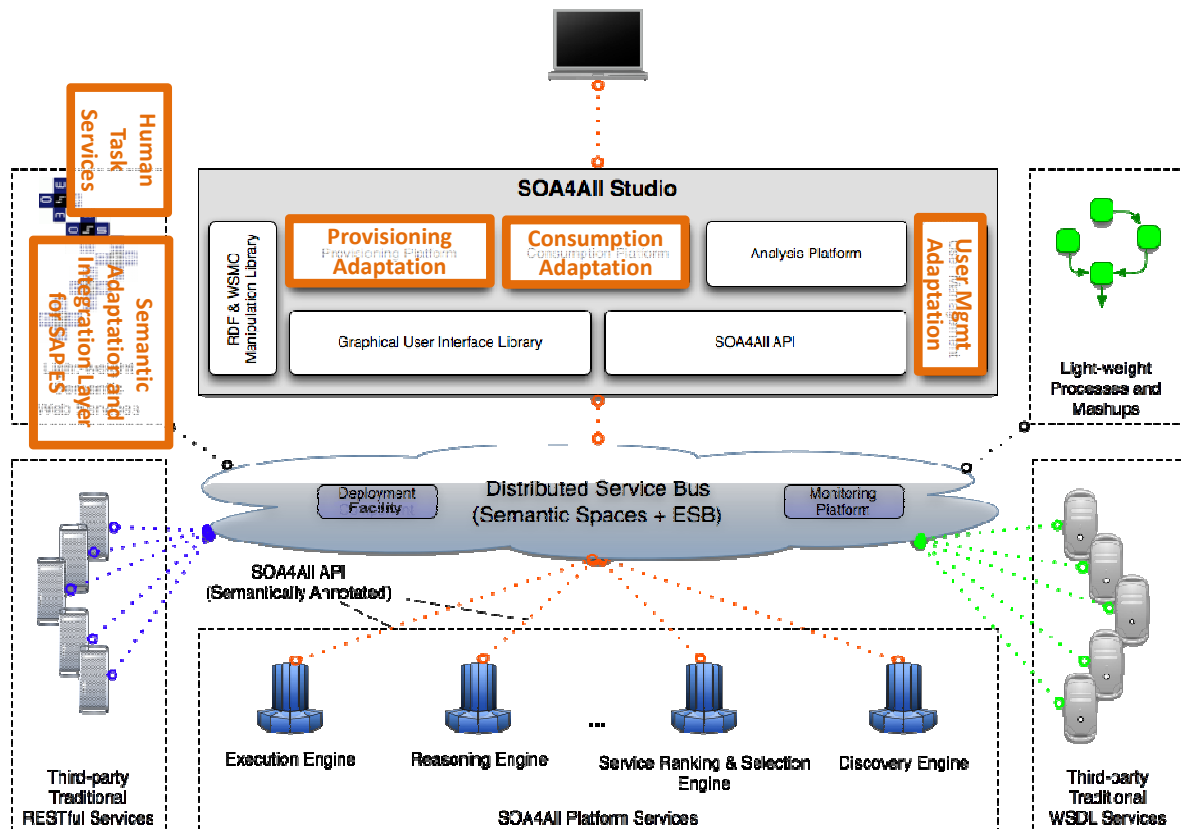


Figure 2: Overview of Adaptations and Extensions to SOA4All

execution of a certain task. Both need to be integrated into the SOA4All Studio. For the management and execution of human tasks at the back end, specific Web services will be provided that are called either by the SOA4All Studio or the SOA4All Execution Engine.

In the following Chapters, we discuss how the SOA4All components (i.e., SOA4All Platform Services, Distributed Service Bus, and SOA4All Studio) will be used in WP7 to develop the envisioned service delivery platform for public administrations as well as the adaptations and extensions to these components that are necessary to fulfill the specific requirements of the use case as described above.

4. Functional and Technical Specification of Components

In the following sections, we detail the use, extension, and adaptation of SOA4All components to develop the service delivery platform of WP7 (see Figure 2). First, we describe the graphical frontend for both civil servants and constituents of a public administration. In Section 4.2, we discuss the management of users and their roles. Section 4.3 describes how services and processes can be searched and selected. The handling of meta-data to facilitate search (amongst others) is designed in Section 4.4. Section 4.5 shows how processes can be modeled by service composer. Section 4.6 describes how the resulting process models are stored. Finally, Sections 4.7 and 4.8 discuss how processes are deployed, executed, and monitored.

4.1 Front End (GUI)

Two user front ends are required to realize the envisioned scenario (see Annex A). The front end for the service delivery platform is intended to be used by civil servants within a public administration with the roles of service composer, service consumer, and service annotator (see Sections 5.2 and 6.2, D7.2). External service consumers as constituents of the public administration will interact with the service delivery platform via an Internet portal.

Front End for the Service Delivery Platform

The dashboard developed in WP2, will provide the entry point to the service delivery platform (see D2.4.1 and Step 1a in Annex A). It will expose the tools of SOA4All and it will provide a menu structure to access them. In addition to the tools planned so far (Profile, Composer, Search, Analysis, and Annotator), additional tools or views may be offered to the target users. For instance, a customized search view will be integrated to let civil servants quickly access pending modification or annotation tasks, which is essentially the result of a search for the corresponding service annotations described in Section 4.4. Relevant use cases for such customized search views are described in Steps 5a/5b and Step 6b of Annex A. Another tool that is required for the public sector scenario is the SOA4All Tasks tool that allows the user to view and execute any pending human tasks which might be assigned to her/him (see Step 14 in Annex A). The list of pending tasks can be retrieved from the business service that will be developed for that purpose as described in Section 5.2. Any extensions to the front end will follow the general design guidelines described in D2.4.1.

Internet Portal for the City of X

The administrative processes that can be modeled and executed by the service delivery platform may resemble public services that the administration offers to its constituents (i.e., citizens or businesses). The execution of such a public service is usually triggered by the constituent filling out an appropriate service request. To allow for a seamless electronic handling of public services (as requested by the EU Services Directive [EC2006]), constituents should be able to issue their service requests electronically.

For demonstration purposes, we will therefore develop a (simple) Internet portal for the City of X that serves as the front end for the constituents of the WP7 scenario (see Step 11a in the Annex A) and that allows to enter the data required for a specific request and to submit the request itself (see Step 11b in Annex A). After submitting the request, the constituent would receive a unique request ID (i.e., a URL) that he can use to check the status of his request.

The implementation of this Internet portal is straightforward and a first version will be available with D7.4. As a basis, a Web server such as the Apache HTTP Server¹ is used that hosts the actual content of the portal. For the planned demonstrations of the project results, we will add a Web pages and forms for scenarios like the one described in the Annex A. The forms for submitting service requests collect the user's input in the browser and forward it to the Web server. The Web server can then call the SOA4All Execution Engine (see D6.5.1) and generate a new HTML page to be displayed as feedback to the user (including the request ID).

The automatic generation of an UI for a specific process is an open research topic and out of the scope of the SOA4All project. For instance, the FP7 project ServFace² aims to enrich service definitions with dedicated semantic information about operations and parameters, as well as process definitions with interaction control and user interface integration details at design time in order to be able to generate automatic UIs at runtime.

4.2 User Management

WP2 provides general so-called Management Services (see D2.4.1) with means for user identification and authentication, resource authorization, auditing/logging, and some simple key/value based user preferences storage.

These services provide the following functionalities:

- **Identification and Authentication:** Identification makes it possible for distinct users to be identified according to their profiles/accounts. Anonymous access is not required for WP7, and a password-based authentication of the identified user accounts is used instead. Ownership of resources (such as services, ontologies, tag clouds, comments and ratings, etc.) is associated with a specific user or group profile.
- **Profile Management:** Essential information associated with each user should be provided at registration time such as: *user id, name, contact e-mail*, and associated *role*.
- **Authorization:** Authorization provides a scheme for defining the access rights to a particular resource per user.
- **Auditing:** Auditing will cover functionality to examine and evaluate the history of changes applied to a resource or the actions performed by a user. Such a change history is also useful to make all user actions in the service delivery platform accountable.
- **Preference Management:** Preference management allows a simple storage and retrieval of user-based and/or application-based settings (key-value pairs). These settings could either be used to personalize settings of the SOA4All Studio tools or to define default values for service interfaces that are retrieved automatically at runtime.
- **Social Graph Support:** The social graph related functionality will make it possible for users to find and add other SOA4All users to their social graph by means of either finding users based on name / e-mail, or to import contacts from existing social networks such as Facebook, MySpace, LinkedIn, OpenSocial, etc. For the use case

¹ <http://httpd.apache.org/>

² <http://www.servface.eu/>

scenario, this feature may be required when the service delivery platform is deployed at an organization where users already participate in such a social network for work purposes.

According to D2.4.1, the Management Services of the SOA4All Studio do not support a dedicated role model with associated access rights as required for WP7 (user roles may be process expert, legal expert, domain expert, clerk, etc., also see Section 6.2 in D7.2). Thus, either the management services will be extended accordingly within T2.4, or WP7 will add an additional management layer on top that uses the profile management to define user roles and a dedicated business service that maps between users and roles and that resolves access rights for a given resource per user or role. Following the interface definitions given in Section 5.2 of D2.4.1, the following additional services will be implemented:

- `setRole(userID:long, role:String):void`
- `getRole(userID:long):String`
- `getUsers(role:String):long[]`
- `hasRole(userID:String, role:String):Boolean`
- `setAccessRight(resourceID:long, role:String, read:Boolean, write:Boolean):void`
- `getAccessRights(resourceID:long, role:String):(read:Boolean, write:Boolean)`

4.3 Service and Process Selection

Service and process selection functions can be accessed by the user either in the dedicated SOA4All search tool of the SOA4All Studio (see Step 1a in the Annex A) or as embedded function of the other Studio tools like the SOA4All Composer. In this section, we describe in more detail how service and process selection is adjusted to the needs of the public sector. As discussed in Section 4.4, processes will have domain-specific attributes such as `ProcessApproval`, `ProcessVersion`, `ChangeHistory`, and `Category`. Hence, we have to extend the search, filtering, and rating mechanisms of SOA4All by features that take into account information about process approval, changes, and categories.

Searching Services and Processes

An important aspect of our use case is service discovery based on full-text. Since most of the users in the public sector are not familiar with sophisticated search engines, a full-text based search is required that crawls all available information about services such as service name and parameters, semantic annotations, tags, categories, and user comments. The “Search & Result Handling” widget that displays the result list from a specific query by the user (as described in D2.4.1) can be extended by the methods

- `setSearchCategory(category:SearchCategory):void`
- `setSearchTag(tag:SearchTag):void`

to allow searching for services within a specific category or with a specific tag. We will further add the functions

- `setChangeTimeframe(startTime:Date, endTime:Date):void`
- `setChangePerformer(userID:String):void`

allowing for the service and process search according to the last change performed and a function

- `setApprovedProcess(Approval:Boolean):void`

allowing to search for approved processes. While the category search is implemented as a full-text based search (see D5.3.1), the change time frame search requires dates as input data. The search for approved processes can be realized by a search on the `ProcessApproval` attribute and will be displayed as a customized search function for users with the appropriate role.

Selecting Goals

The SOA4All Studio will provide designated tools for supporting the user to select a goal that is then automatically linked to one or more concrete services (e.g., see Step 7e in Annex A). As described in D2.2.1, the “Template Completion” component will help the user in finding an appropriate goal and in specifying missing information, the NLP (natural language processing) component allows the user to find goals based on descriptions typed in textually using natural language, and the “Incremental Goal Revealing” component will allow users to define goals in several iterations getting more detailed with each step. Concrete method interfaces to these components will be defined by T2.2 at a later stage of the project.

Filtering Services and Processes

The Client-Side Filtering widget (as described in D2.4.1) will be extended by a category, tag, process change, and approval filtering functionality. Similar to the search, we will extend this widget by the methods described above allowing setting the `Category`, `Tag`, `Version`, and `ProcessApproval` for the filter. We will integrate another filtering functionality that is based on the approval context information. If the value of the approval attribute of a process is `noApprovalRequired`, the legal expert will not see them in his default view after logging in to the Studio (see Step 5b in Annex A). Moreover, all users will be provided with filters that display (1) the processes and services they use most frequently, (2) the services and processes they have used most recently, (3) and processes which have been changed recently in the favorites list of the Studio UI (see Step 1c in Annex A).

Ratings and Rankings

The Rating widget as described in D2.4.1 needs to be extended in order to describe how useful a service is for a certain category. We will provide an extra icon for the rating result according to the category. Furthermore, we will use the standard SOA4All rankings based on security, availability, reliability, provider trust etc. The recommender system developed in T2.7 will be used to sort result lists generated in a search according to the estimated relevance for the user.

4.4 Annotations and Community Support

The processes are stored in a shared repository (see Section 4.6). In SOA4All, this shared repository is realized by a distributed Semantic Space (see D1.3.2A). The processes are also semantically described. The user creates some annotations for the process as a whole using the annotation editor developed in the service provisioning Task 2.1 (see D2.1.2), which might be embedded into the SOA4All Composer (see D2.6.1) via the SOA4All Dashboard (see D2.4.1). The standard annotations within SOA4All cover input and output data, preconditions, effects, and capabilities. Additional annotations are currently under investigation.

Within WP7, we will define specific annotations for processes that are important in the public sector. For instance, as described in Chapter 6 of D7.2, the public sector scenario requires a compliance check by a legal expert before a new process is allowed to be executed. This procedure can be implemented by using a designated process attribute named `ProcessApproval`. The attribute `ProcessApproval` serves as an indicator for the compliance check according to business and legal rules. The default value of `ProcessApproval` when creating a new process is `notApproved`. Further values should be `rejected`, `accepted`, or `noApprovalRequired`. The attribute can be changed only by a user with the appropriate rights (see Section 4.2) and is checked before the deployment and execution of a process (see Section 4.7).

An important aspect is to support users in organizing and finding processes. For the public sector, services and processes are assigned different categories such as the topic (e.g., human resources, identification cards, car registration etc.) or the department (finance, personnel etc.). Hence another attribute named `Category` is needed that can store several pre-defined categories.

Furthermore, we will provide means to document changes on each process model. This can be done by storing the relevant information about the change in terms of author and timestamp in the attribute `Version`.

In order to support the reuse of processes via a shared process repository that is managed within a public administration such as in the fictional City of X, we will provide annotation mechanisms for comments and ratings that can then be used by the search and filtering mechanisms (see Section 4.3) and the recommender system developed in Task 2.7. We will thus create the process attributes `Comment` and `Rating` that are time-stamped and referenced with the respective `userID`. The `Comment` attributes will provide ordinary text fields (see Figure 4a). As depicted in Figure 4b, the `Rating` attribute is composed of different aspects. As mentioned above, we will use the standard SOA4All rankings based on security, availability, reliability, provider trust etc. The process commenting functionality will again be linked to the user model defined in Section 4.2. In our scenario description, Claudia

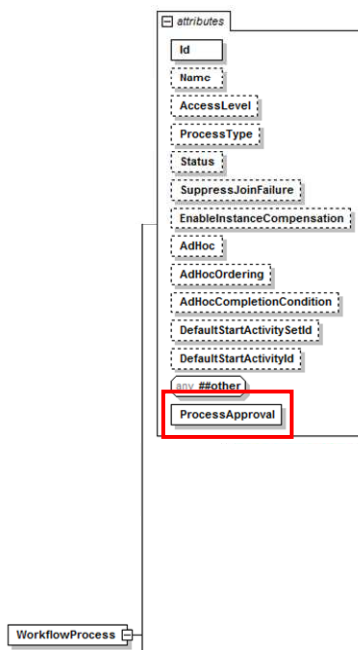


Figure 4a: Extended Attribute List for Process Annotations

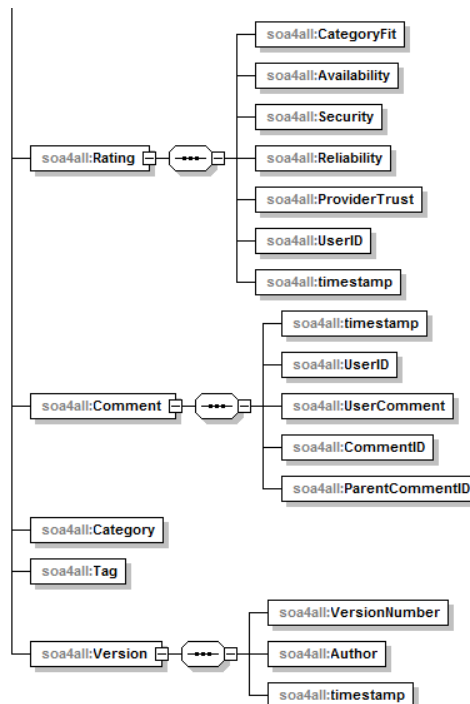


Figure 4b: Extended Element List

will comment her process approval (see Steps 5b and 5c in Annex A). By linking the commenting functionality to the user model we will make sure that only legal experts are allowed to change the approval attribute (see attribute `ProcessApproval` in Figure 4a).

Further details of the service annotation and categorization approach will be given in D7.6.

4.5 Process Modeling

As described in D7.2, civil servants usually have a very detailed understanding about processes but often do not have the skills to use formal BPM tools. Thus, to support these users in modeling processes, SOA4All will

- (1) provide a lightweight modeling language (WP6)
- (2) apply semantic technologies to support the discovery (WP3 and WP5) and composition of services and processes (WP6) and to hide details via goals (WP2 and WP6)
- (3) offer a user-friendly GUI with wizards to guide the user in modeling tasks (WP2).

In this section, we present an overview of how to customize the SOA4All Composer to meet the requirements of the WP7 use case.

Customizing the SOA4All Composer

Civil servants will use the SOA4All Composer to model processes (see D2.6.1). Step 1f in Annex A shows a screenshot of the SOA4All Composer UI during modeling with active Modeling Canvas and Tools components. Experienced users may model a new process in the so-called “free mode” by placing the graphical elements of the modeling language (start process, stop process, activity, and connector) via drag and drop on the canvas. Details of an activity (e.g., service parameters) can be specified in a details box displayed for the selected activity.

Activities can either be concrete services, goals in case the user is not able to provide any information required to specify a concrete service, or human tasks. We will extend the editor by an activity attribute stating whether the activity represents a human task. This information will be used by an extension of the process execution engine provided by T6.5 that handles machine-based and human tasks.

The locking of certain process parts is quite important in the public sector in order to prevent modifications of parts that are modeled according to legal regulations and should not be changed or to prevent modifications by unauthorized personnel. Within WP7, the SOA4All Composer is thus extended by a locking functionality for activities and connectors. This functionality will be based on a user model defining access rights per activity and connection types. We will thus extend the attribute list of activities by an attribute `AccessRight`. The `AccessRight` attribute will provide references to appropriate attributes in the user model defined in Section 4.2. Checks will be executed on the client-side of the SOA4All Composer (see D2.6.1) before allowing a modification by user input. Feedback on the locking status of certain elements will be provided visually with an appropriate “locked” icon that is displayed on top of the each locked element.

Domain-specific Wizards for the SOA4All Composer

The extensions of the SOA4All Composer performed in the context of WP7 include smart wizards. For instance, as described in Step 7 in Annex A, a smart, domain-specific wizard `AddCreditCardPayment` is implemented that contains a control flow to describe the stepwise guidance as well as a description of the actual modification to be performed at each

step. The concrete implementation of smart wizards is not yet defined but will be addressed by T2.6 in the remainder of the project.

Domain-specific Process Templates

As described in D6.3.1, there is wide agreement that process patterns and templates can accelerate the process of designing a solution and reduce modeling time. The templates can be generally applicable or domain-specific. The SOA4All Composer will allow for the creation of both kinds of templates. In the context of WP7, we will develop several process templates for the public sector. The actual modeling of a template does not differ from the modeling a regular process. We will rely on an updated version of the lightweight process modeling language (T6.3) and of the Composer (T2.6) for a specific handling of templates.

4.6 Process Storage

All processes created with the SOA4All Composer are stored to the shared repository (see Step 4 in Annex A and D2.6.1) that is accessible to all users within a public administration, which have a login and the appropriate access rights (see Section 4.2). After modifying an existing process, a new version is created and also placed in the repository (see Step 8 in Annex A). Thus, civil servants exchange their business knowledge via the process models and the attached annotations described in Section 4.4.

The shared repository is typically hosted either by the public administration or by a 3rd party platform provider (see Section 5.2, D7.2) and is accessible within the administration's Intranet. However, several administrations could also exchange processes (and therefore business knowledge) by either connecting their repositories or by running a single instance. Again, access to the shared repository and to selected processes in such an inter-organizational setting can be controlled via the general authentication of the service delivery platform and by the individual access rights per user.

SOA4All provides two types of storage services:

- WP1 provides a Semantic Space infrastructure as a distributed storage for semantic information encoded in RDF (see D1.3.2A)
- In order to simplify the use of the Semantic Space, WP2 provides the Storage Services, an easy to use API that provides simple storage functionalities (see D2.4.1). Those services consist of fundamental functionalities that enable RDF creation, update, deletion and querying (SPARQL) functionalities. Also, the WP2 storage services allow the storage of files such as WSDLs.

The SOA4All Composer builds upon the WP2 storage services (see D2.6.1) to store the processes as well as the semantic annotations described in Section 4.4.

4.7 Process Deployment and Execution

In this section, we describe how a particular process can be deployed and executed by a user of the service delivery platform (see Step 10 in the Annex A). The process deployment and execution environment provides self-adaptation at design time to make use of the specific capabilities of component services or at runtime to adapt to contextual requirements (such as the selection of the fitting payment service from the "Handle Payment" goal described in Step 7e and Step 14 in Annex A) and unforeseen faults (see D6.5.1). Once a lightweight process has been designed, it needs to be translated into an executable process by the so-called Artifacts Generator developed in T6.5 (see Figure 5 and D6.5.1). The

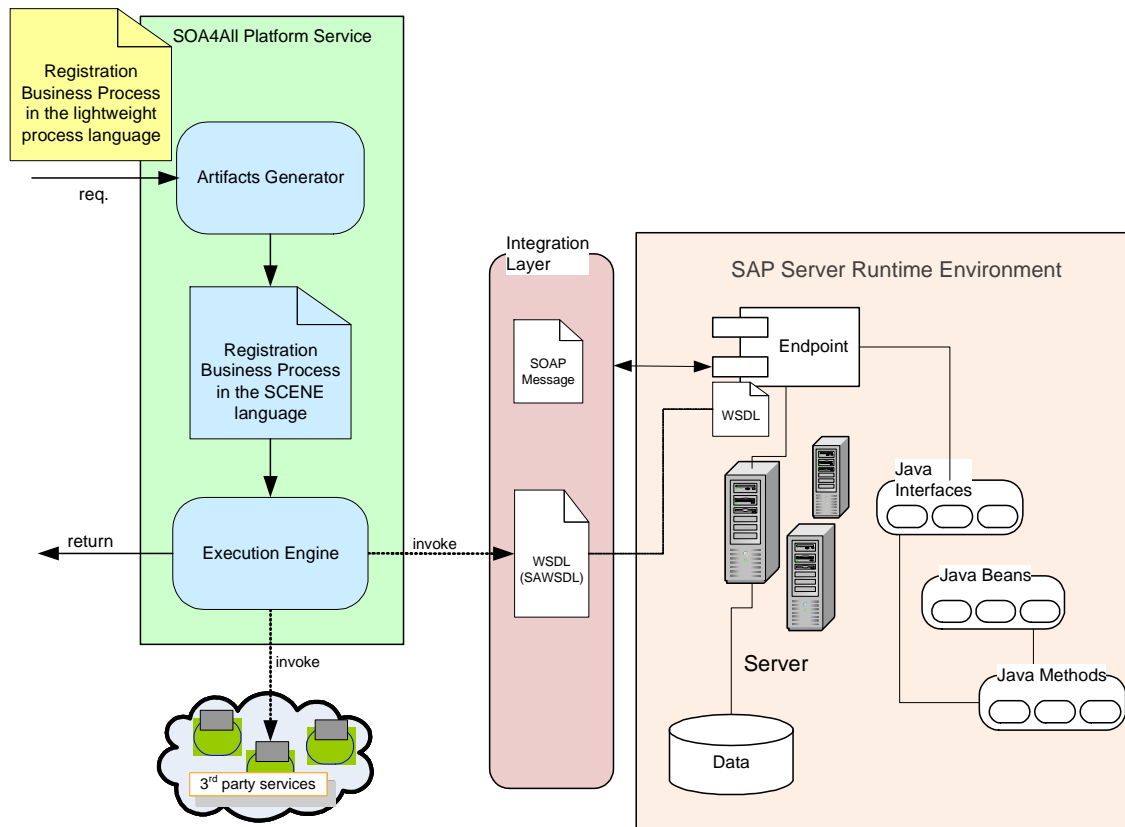


Figure 5: Process Deployment and Execution

artifacts to be generated include the process described in an executable language that can then be deployed to the Execution Engine developed in T6.5. This executable language is composed of three main parts:

- a BPEL weaved for decoupling the BPEL process from the adaptation and reconfiguration logic,
- a set of rules for implementing the policies that enable self-adaptation and self-reconfiguration at runtime,
- a set of mapping scripts for solving possible mismatches in service interfaces.

In SOA4All, services involved in the composition are semantically annotated using the grounding schema described in D1.2.1. An annotation of a Web service is provided using a common domain-specific ontology. Semantically-annotated WSDLs of a concrete Web service also describe all the data types used as input and output parameters of the operations and operation names (also see Chapter 6). This information is used for automatically generating the mapping scripts. Semantic annotations for the services used (see Chapter 5) will be provided with D7.6. The result of a process execution is returned to the user and can be accessed via the SOA4All Analysis tool described in the following section.

4.8 Process Monitoring

The monitoring and management tool suite of the SOA4All Studio provides information about executed services and processes for service consumers, composers, annotators, and

providers (see D2.3.1). Service consumers can check the status of all running process instances. For example in the use case scenario, an authorized civil servant can check pending requests of the “Registration of a Business” process (see Step 13 in the Annex A). A user can also access a more detailed view of the status of a certain process instance and its included services in the form of an overlay to the original process model (see Figure 25, D2.3.1). Service composers can view Quality of Service (QoS) data such as the average response time or the availability rate to decide among alternative Web services. Moreover, alerts can be defined that trigger a notification in case QoS parameters are violated (see D2.3.1).

For a service provider with direct access to the service delivery platform (assuming a contractual agreement with the public administration), further monitoring information can be accessed: the number of unique users, the origin of users, the minimum and maximum numbers of concurrent calls, list of processes that call a particular service etc. A direct access to the service delivery platform would also allow the service provider (e.g., SAP) to quickly react to quality of service issues and to seamlessly deploy new or modified services.

5. Selected Services for the Demonstrator

One major objective of the SOA4All use case of WP7 is to demonstrate the usage of both public Web services and SAP's Enterprise Services (short: SAP ES) with a special focus on enterprise system functionalities. The following sections will discuss the Web services, services for handling human tasks, and SAP ES to be used within WP7 in order to give a better insight into their provided functionalities, usage, and business objects (if available).

5.1 Public Services

For the underlying process of a registration of a business (see Figure 16, D7.2), the following two Web services, which are provided by the Web service platform Webservice.com [Webservice.com2009], will be incorporated. For this purpose, they will also be annotated semantically (see Chapter 6).

Credit Card Validity Check

During the first steps of the process, the civil servant files the case of a new registration. If the user has declared to be charged via a credit card payment, the respective credit card has to be checked for its validity by the servant. For this purpose, a 3rd party Web service called "ValidateCreditCard" will be invoked, which is capable to check credit cards of Master Card, VISA, Amex, and DINERS as illustrated in Figure 6. The service requires type and number of the credit card to be checked. Its response is a Boolean data type for indicating the validity. The complete WSDL of this Web service can be found at:

<http://www.webservicex.net/CreditCard.asmx?wsdl>

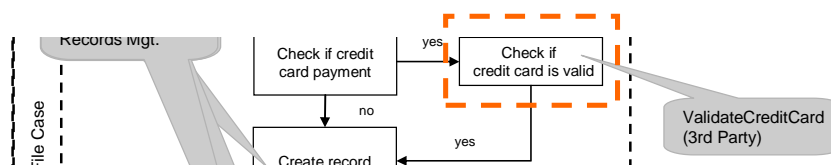


Figure 6: Invocation of the Web Service ValidateCreditCard within the process "Registration of a Business" V2

Address Validity Check

In order to avoid any malpractice of the user interfaces provided by the public administration of the city and to guarantee a comparatively high data quality, the address of the applicant

will be automatically checked via a Web service such as “UK Location”³. Figure 7 shows the point of invocation within the execution of the process. For the verification task, the method `ValidateUKAddress` will be used within the Web service. For invoking the service, the town, county, and postal code are required in order to fulfill the checking task. The result is a Boolean data type. A detailed WSDL description with all provided methods, parameters, and results is available at:

<http://www.webservices.net/uklocation.asmx?wsdl>

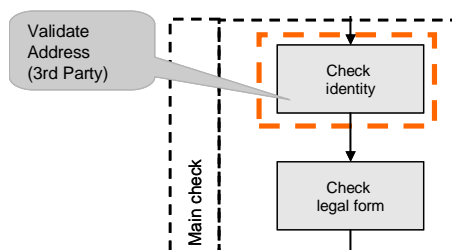


Figure 7: Invocation of the Web Service `ValidateUKAddress` within the process “Registration of a Business” V2

5.2 Services for Handling Human Tasks

In the public sector (and other domains), processes may contain activities that are human tasks. For instance, the activity “Check legal form” as depicted in Figure 7 is a human task to be executed by a civil servant. Instead of requiring a fully-fledged BPEL for People [BPEL4People2005] Extension of the SOA4All Execution Engine, which is out of the scope of the SOA4All project, we intend to develop a dedicated *Task Server* that provides several services to enable the handling of human tasks via services (see Figure 2):

- `registerHumanTaskCallback(userID:long, processURI:String, activityURI:String) : void`

If the SOA4All Execution Engine executes the process with the URI `processURI` and reaches the activity with the URI `activityURI` that resembles a human task, it calls the operation `registerHumanTaskCallback` asynchronously to register the user with ID `userID` as the responsible user for the activity `activityURI` of the process

³ “UK Location” handles addresses in the UK only. However, the service is used as a proof-of-concept and can easily be replaced by services with the same functionality, which are suited for the City of X.

processURI at the Task Server.

- `executeHumanTask(userID:long, processURI:String, activityURI:String, result:Boolean) : void`

This service is called from the front end side (i.e., the SOA4All Tasks tool of the SOA4All Studio) when the user has executed the task. In the simplest case (e.g., for an approval or check), the user can either decide “Rejected”/“Failed” or “Approved”/“Successful” as a false or true result value.

- `executeHumanTaskCallback(userID:long, processURI:String, activityURI:String, result:Boolean) : void`

The Task Server notifies the Execution Engine as soon as the user with `userID` has actually executed the task of the activity `activityURI` for the process `processURI` with `result true` if the task was executed successfully, else with `false`, so that the Execution Engine can trigger any follow-up activity in the process model.

- `getTasks(userID:long) : (processURI:String, activityURI:String) []`

Retrieves all pending tasks for a certain user, e.g., for displaying a task list to the user.

- `cancelTask(userID:long, processURI:String, activityURI:String) : Boolean`

Cancels a pending task of a user for a certain process and aborts the process. It returns `true` on success, `false` on failure.

- `cancelTasks(userID:long) : Boolean`

Cancels all pending tasks of a user and aborts all corresponding processes. It returns `true` on success, `false` on failure.

All services will be accessible as WSDL and WSMO-Lite (see D3.4.2) services. This approach allows us to use the Execution Engine as designed by T6.5 without further specifications. The Task Server could also take over more sophisticated business logic like determining an appropriate user dynamically at runtime from a given role (instead of having static users as indicated above). The necessary specifications of the service parameters are done at design time using the SOA4All Composer. As described in Section 4.1, the SOA4All Studio will also provide a Tasks tool for users to access and perform their designated human tasks via the service interface listed above.

5.3 SAP Enterprise Services

First, we give a general introduction to SAP Enterprise Services and describe their technical foundation. In Section 5.3.2, we describe a set of services that provide the business functionalities required to realize the sample scenario of this use case.

5.3.1 General SAP Architecture, Data Models and Access

In order to realize benefits of a service oriented architecture, SAP has developed a strategy to provide services of nearly all SAP products. By means of this concept, SAP envisages for its customers an increased reuse rate of software components [Endrei2004], a reduction of complexity in terms of decomposing large applications into single software functionalities, a faster time to market due to the easier development and deployment effort [Bieberstein2005],

as well as the support for business process automation and business process innovation [SAP2008].

Web services provided by SAP are called Enterprise Services (ES) and encapsulate functions of SAP business software, e.g., from SAP ERP 6.0. Enterprise Services can be managed and accessed via the SAP NetWeaver SOA Platform [SAP2007]. The design of these ES follows the concepts of generally accepted Web services standards such as WSDL. In comparison to the majority of the services that can currently be found on the (public) Web, ES can be differentiated by the following characteristics:

- *Business semantics:* ES use a harmonized enterprise data model with so-called global data types (GDTs), e.g. Date which is specified as YYYY-MM-DD. For a comprehensive overview and description of all business objects used by ES please refer to the SAP Data Type Catalog listing Core Data Types and Global Data Types: <https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/303fd192-1db2-2a10-59b9-9dfd93f4b10f>
- *Backward compatibility and reuse:* ES provided by SAP are compatible with older versions of services and can be reused across all SAP solutions
- *Standards:* WSDLs are described and created by GDTs based on the document standard UN/CEFACT CCTS (Core Component Technical Specification) [UN/CEFACT2007]

SAP considers ES as representatives of a “common language of business” [SAP2007]. Figure 8 illustrates the overall architecture encompassing processes, ES, and enterprise systems. Single functionalities from SAP enterprise systems (e.g., customer relationship management CRM) are exposed and organized as ES. These services are deployed on an SAP NetWeaver environment and are coupled with single process activities to fulfill the needs of business processes.

For more detailed information about SAP's concept of ES, the following introductions and documentations can be helpful:

- Enterprise SOA in a Nutshell: gives a brief overview including a description of all relevant components of SAP's ES
<https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/c019bf26-8bb8-2910-4f8f-e9bd55eda650>
- How to Consume ES Workplace ES in Visual Composer
<https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/a160392c-0e01-0010-7784-9cc564d871d2>
- ES Workplace Handbook: describes how to search and access ES via the central platform ES Workplace
<https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/d91e4d16-0b01-0010-57b5-c4473111136f>
- ES Documentations: very detailed information including transactions, maintenance, consumption of ES, etc. categorized by process components of ERP, SCM, SRM, and CRM
<https://www.sdn.sap.com/irj/sgn/explore-es?rid=/webcontent/uuid/c0cd8360-3b74-2910-0fae-dcceed7328e7>

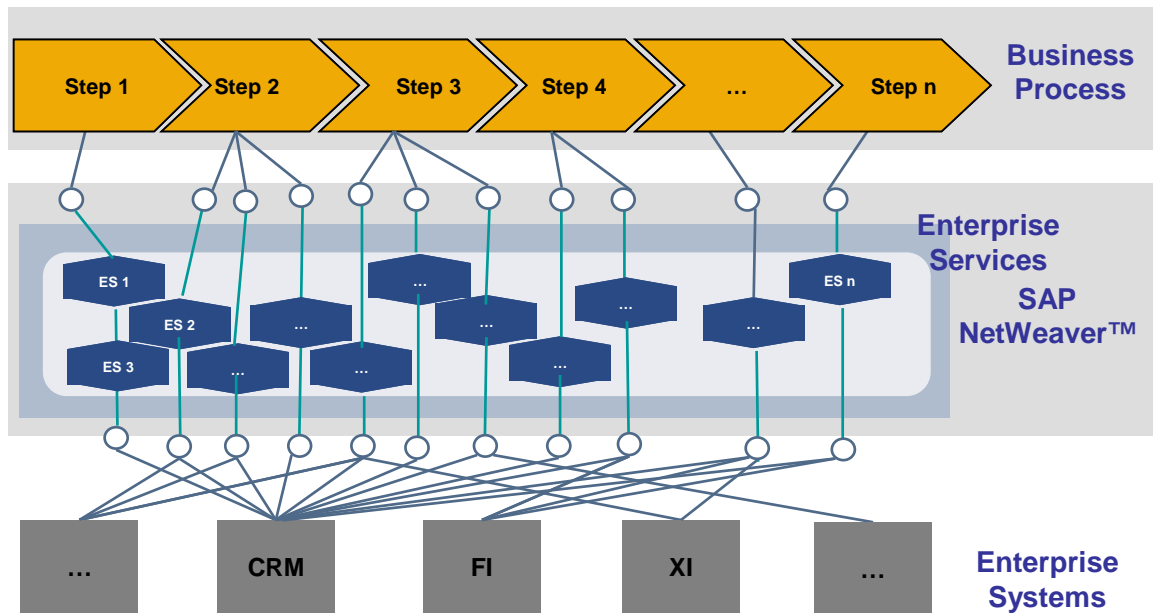


Figure 8: Overall SAP NetWeaver Architecture with Processes, ES, and Enterprise Systems (in the style of [König2004])

For establishing an “exemplary enterprise SOA information and evaluation environment”, SAP provides “a central place to view consolidated information of all available ES delivered by SAP” [SAP2008] – the ES Workplace.

Customers, developers, and business partners can search via the ES workplace for ES and can directly access them through the NetWeaver Developer Studio or (for test purposes) via the browser. The ES Workplace allows access to “the latest versions of all ES available for the SAP Business Suite 2005 in a hosted environment” [SAP2008]. ES, which are published in the ES Workplace, cover functionalities of the following systems of the SAP Business Suite: Enterprise Resource Planning (ERP), Exchange Infrastructure (XI), Supply Chain Management (SCM), and Enterprise Portal (EP). All ES in the ES Workplace have been modeled and developed by SAP architects and developers and are stored in the ES Repository (ESR).

For accessing ES different views are provided, which facilitate the search and selection process. Technical experts can browse through the ES Index, whereas users with less technical background can search within the SAP solutions maps for different industries (e.g., banking, insurance, automotive etc.) that represent core functionalities, organizational units, and processes of the industry of interest.

So-called ES Bundles package ES into “main business scenarios and processes as well as industries” [SAP2007]. In addition to the packaged ES an ES bundle contains:

- the documentation how to use the relevant services for extending and reconfiguring specific business processes
- an explanation of the relevant processes, business scenarios and roles involved
- descriptions of used business objects

The ES Bundles are organized and stored in the ES Bundles catalogue which can be accessed via the ES Wiki:

<https://wiki.sdn.sap.com/wiki/display/ESpackages>.

From an end-user viewpoint ES are comparatively complex to be consumed. As mentioned above, they can be accessed either via the NetWeaver Developer Studio (addressing technical experts) or for test purposes also via the WS Workplace, or to be more precisely via the WS Navigator. The WS Navigator provides a browser-based user interface for invoking services. However, due to the complexity of the data types and the ability of the underlying functions to be customized, preconditions and mandatory parameters have to be considered and specified by the user. As shown by the list of parameters for the Enterprise Service “Create Public Sector Document” on the left side of Figure 9, the usage of ES would require from end users a detailed technical knowledge about business objects dependencies for example. Consequently, one of the main objectives of WP7 is to facilitate the integration and application of ES for end users daily work.

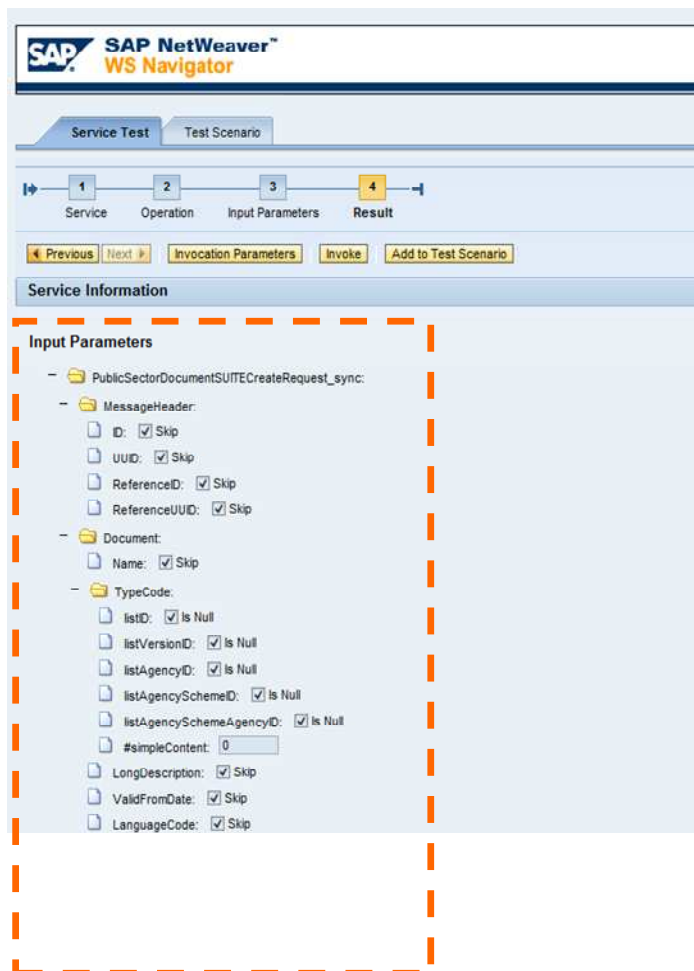


Figure 9: WS Navigator for Enterprise Service “Create Public Sector Document”

5.3.2 ES Bundle: Records and Document Management

One of the aforementioned ES Bundles is the “Records and Document Management” bundle. The ES stored in this bundle have a high relevance for the use case scenario of WP7 and are used for the planned Demonstrator. The use case of WP7 demonstrates the process of a registration of a business in a German municipality, which imposes the generation of several documents that are assigned to a record file representing every registration case (see

Chapter 6, D7.2). The ES Bundle “Records and Document Management” provides functionalities for reading, creating, and changing documents and record files as well as managing relations between documents and record files. All ES of this bundle are based on the enterprise system SAP Records Management that meets all relevant requirements of a document management system such as storing documents, managing meta-data of documents etc.

In particular public sector organizations work with large amounts of documents and therefore rely on proper document management mechanisms for their business processes. By means of the ES of this bundle a variety of document MIME types (e.g. emails, text documents, spreadsheets, forms, MP3 recordings etc.) can be stored and managed.

The benefit of the service oriented approach in this context is the comparatively flexible integration of the document management services in document processing systems or scanning tools which allows an automatic storing and management procedure of the respective documents and their record folders. Also, the integration in email clients enables the organization-wide management of emails or even mail folders.

Without a service-oriented approach, integration of such functionalities into existing information systems would have required a significant effort.

Key Business Objects

The key business objects of this ES bundle are *documents* and *record folders* (previously also mentioned as record files). A variety of binary objects of any MIME type can be considered as document. These documents are assigned to record folders which can be specified with additional attributes supporting, e.g., restrictions and legal implications of business processes.

List of Concrete Services

Within the ES bundle, the following services are available for reading, creating, and updating operations:

- Create Public Sector Document (for creating the business object “document”)
- Create Record Folder (for creating the business object “record folder”)
- Find Public Sector Document by Elements
- Find Record Folder by Elements
- Read Public Sector Document
- Read Record Folder
- Read Record Folder Business Folder Business Object Reference
- Update Public Sector Document
- Update Record Folder
- Update Record Folder Business Folder Business Object Reference (for attaching a document to a record folder)

The SAP ES Wiki describes several use cases for a more comprehensive overview of the ES Bundle Records and Documents Management such as “Storing Email in SAP Records Management via Email Client”, “Building Permit Request”, and “Create a Logistical Assessment Report”. The second one is strongly related with the use case shown in WP7 which encompasses also a permit procedure generating several documents. A typical sequence of services to be invoked would be as follows:

1. Create Record Folder: the required record folder will be generated and specified
2. Create Public Sector Document: the required document will be generated and specified
3. Update Record Folder Business Folder Business Object Reference: the document will be assigned to the respective record folder

For further information about this ES Bundle please refer to:

<https://www.sdn.sap.com/irj/scn/wiki?path=/display/ESpackages/Records+and+Document+Management>

6. Semantic Adaptation and Integration Layer for SAP ES

In this chapter, we present a concept on how to integrate SAP ES into the SOA4All architecture so that they can be consumed by business users via the SOA4All Studio as described in Annex A. The existing SAP ES are SOAP [SOAP2007] Web services with syntactic descriptions of the service interfaces in WSDL [WSDL2001]. Consequently, additional semantic descriptions need to be provided in WSMO-Lite to allow their use within SOA4All. As specified in D3.4.2, WSMO-Lite is a lightweight set of semantic service descriptions for WSDL-based Web services using appropriate domain ontologies (that will be provided with D7.6). WSMO-Lite defines the

- *information model* for the input, output, and fault messages of a service, the
- *functional descriptions* in terms of conditions that must be satisfied before a service can be invoked, effects that hold after a service was invoked, and functionality classifications for categorizing a service's functions, and the
- *nonfunctional descriptions* for additional properties of a service (security, pricing, QoS etc.)

These domain ontologies can be expressed in any W3C-compliant language that uses RDF syntax such as RDFS, OWL, or WSML. With the SAWSDL [SAWSDL2007] annotation mechanism the original WSDL service interface of an ES will be extended to link these semantic descriptions with the respective syntactic parts of the service interface.

Because existing SAP ES will be used to implement the WP7 scenario and their WSDL interfaces cannot be changed directly at the service endpoint, we propose using a *semantic adaptation and integration layer* inspired by the Web service broker approach presented in [Alur2003] (see Figure 2). This layer incorporates the following main functions:

1. reuse and expose existing SAP ES without modifying their WSDL interfaces,
2. reduce the complexity of invocation of ES by handling optional or default parameters transparently,
3. bridge the gap between SAP ES and requirements of semantic service provisioning in the SOA4All context,
4. decouple the semantic annotations from existing SAP ES, thus helping to mediate transparently between the requirements of Semantic Web services (at the semantic level) and the actual invocation of these classic ES (at the invocation level)⁴,
5. provide necessary application-specific message transformations.

As a short introduction to the topic of designing an integration layer, we will first describe the state-of-the-art architecture for WSDL-based services and briefly discuss the SOA4All approach to add a semantic layer on top. The actual contribution of WP7 is the design of the semantic integration layer for the adaptation of SAP ES, which is given in Section 6.3.

⁴ See Figure 1, D1.2.1.

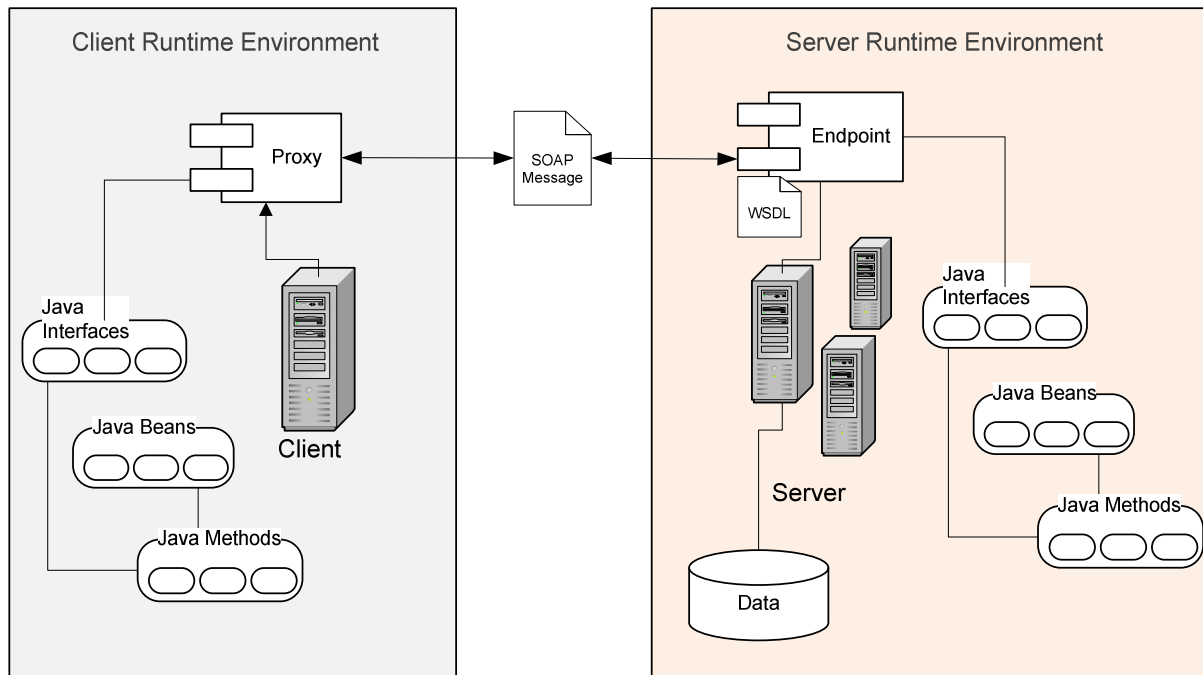


Figure 10: Classic Web Service Architecture

6.1 Classic Web Service Architecture

In the classic Web service architecture as depicted in Figure 10, the WSDL service description defines a service contract of the underlying SOAP Web service. It specifies what operations are offered by the Web service as well as the required data types and exchanged message format, etc. On the server-side, the server runtime environment is responsible for provisioning the Web service. It consists of a service endpoint which listens for SOAP request messages transmitted via the underlying transport protocol such as HTTP. The runtime environment determines the target service indicated in the request message, i.e., which WSDL operation the message is intended to invoke. Consequently, it determines which concrete corresponding method implementation must be invoked on the server-side.

The target method is defined in an interface, called a service endpoint interface (SEI), which is a constituent part of the server-side Web service system. The concrete implementation of the defined method, for instance, is provided in a Java class which implements this service endpoint interface⁵. The de-serialization subsystem converts the invocation parameters of the SOAP message into instances of corresponding Java classes. This conversion process, conventionally called *unmarshalling*, is controlled by predefined binding information that maps instances of XML Schema types, i.e. instances of XML documents that conform to the XML Schema type definition, to the corresponding Java objects. It passes them to the Java method target for invocation. Once the method invocation has finished, the return object is converted or *marshalled* back to the corresponding instances of XML type by the serialization subsystem. The runtime environment encapsulates the return type as XML in the SOAP response message, which conforms to the WSDL response message format definition.

On the client-side, a service endpoint interface (SEI) can be generated from the WSDL service description document. The SEI consists of the methods which represents a client-

⁵ For the remainder of this Chapter, we assume a Java-based implementation.

side view of the server-side methods provided by the Web service. The SEI must be implemented by a proxy in the client-side runtime environment. Based on the actual service method to invoke, the proxy is responsible for the generation of SOAP request messages at runtime according to the message format defined by the WSDL interface. The SOAP request message consists of the necessary invocation parameters. The runtime environment serializes the Java objects into instances of XML Schema types and encapsulates them into the SOAP request message.

The proxy sends the SOAP request message over a transport protocol binding to a Web service endpoint. When the Web service returns a SOAP response message, the proxy is responsible for interpreting and handling the response message on behalf of the client. In case of an exception or error indicated by a SOAP fault message, it invokes the error handling code on the client-side. Subsequently the proxy passes the message to the deserialization subsystem which processes the response message reversely, i.e., taking the return type and instantiating the corresponding Java bean object on the client side.

6.2 Semantically-Annotated Web Services

In the context of SOA4All, Web services are semantically annotated in order to support, e.g., automatic service discovery, selection, composition, and service invocation. We use WSMO-Lite (see D3.4.2 and D1.2.1) for the purpose of providing a light-weight annotation and grounding approach. WSMO-Lite utilizes SAWSDL to provide references to domain-specific ontologies by augmenting WSDL service description using the `modelReference` attribute (see [SAWSDL2007]). SAWSDL also provides a concept of lifting via the `liftingSchemaMapping` attribute and lowering via the `loweringSchemaMapping` attribute to transform between syntactic XML Schema data types and their semantic ontology representations. This latter feature is called grounding (see D1.2.1). Different ontology formalizations such as RDF [RDF2004], RDFS [McBride2004], OWL [Dean2004] and WSML [Steinmetz2008] are suitable as formalized representations of the semantic model. Moreover, other WSDL parts such as the messages, operations, and service elements etc., can also be annotated and grounded in a similar manner using SAWSDL.

An example of the usage of the WSMO-Lite grounding mechanism with SAWSDL for a typical credit card checking Web service is provided in Listing 1 of Annex B. In the listing, the SOAP request element `individualCreditCheckRequest` represents a request message wrapper element to be sent to the Web service. This element is used in the incoming SOAP request message for invocation at runtime. The attribute `modelReference` is used here to indicate three references to existing definitions of ontology vocabularies defined for this service. These references are identified by their URI references (consisting of the namespace part and the local part to identify definitions uniquely) to their corresponding vocabulary definitions, i.e. `DeltavistaUser`, `CurrentTime`, and `Individual`. Since the incoming request SOAP message uses instances of XML Schema types to represent these constituent elements of a valid request message, their corresponding vocabularies at the semantic level must be *lowered* to these XML type instances (see D1.2.1, Section 4). The `loweringSchemaMapping` attribute is used in this regard to reference a transformation service (indicated here by the XSPARQL [Polleres2009] service URI reference). After the Web service has terminated, its response message is transformed back, i.e., *lifted* to the corresponding ontology concept at the semantic level. The `liftingSchemaMapping` references an XSPARQL lifting service to perform this operation.

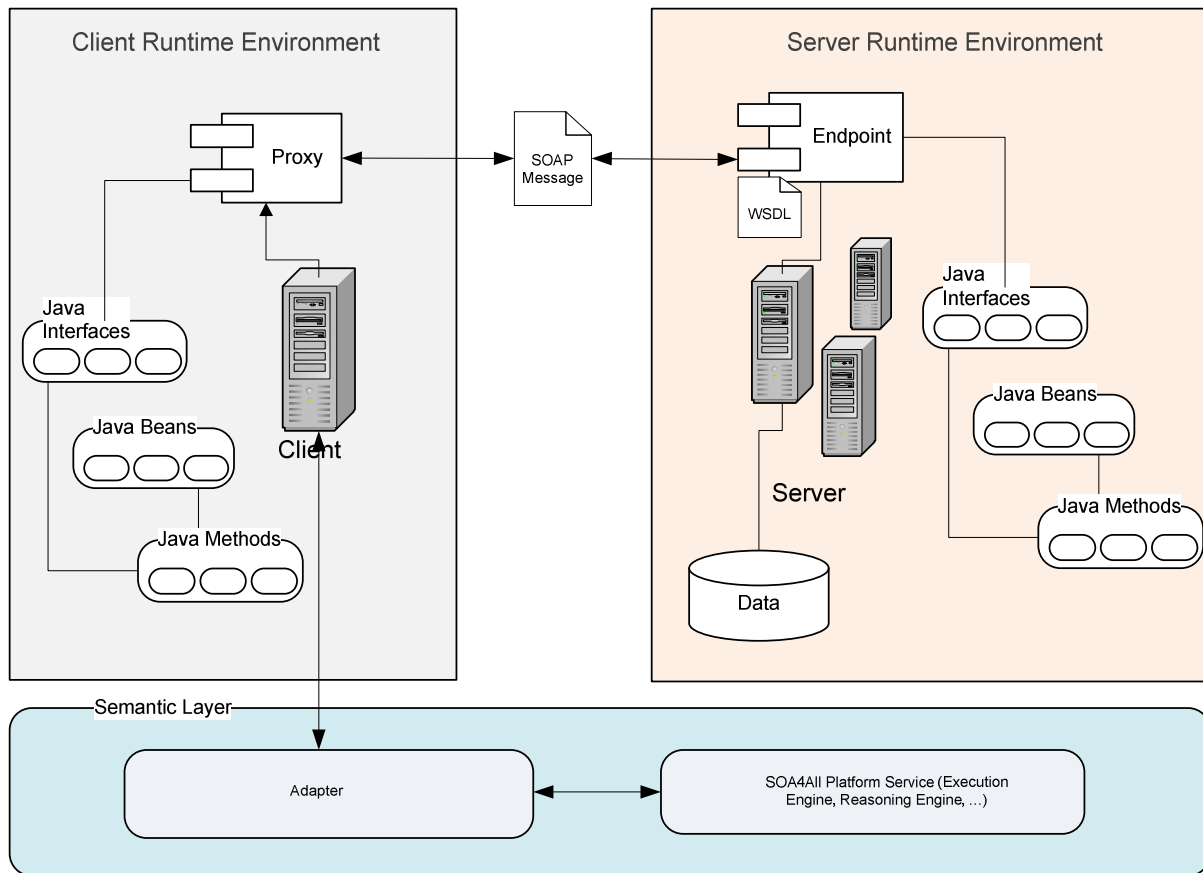


Figure 11: Web Service Architecture with Semantic Layer

A semantic layer is necessary to enable semantic adaptation of the classic Web service architecture, see Figure 11. The semantic layer consists of an infrastructure of the SOA4All Platform Services including a Service Registry to store the semantic service annotations, a semantic service discovery engine, a service ranking and selection engine, a reasoning engine, and a process execution engine (see Figure 1 and D1.4.1A).

A Web service consumer may formulate his service requirements in WSMO *Goals* [Roman2007]. These requirements represent the user view of application-specific capabilities on the desired Web service(s). A goal demands fulfillment of consumer requirements on an abstract level using WSML [Steinmetz2008] to express capabilities in post conditions via axioms and logical expressions. Referring to the handle payment process (see Figure 17 in D7.2), one of the operations to handle a credit card payment is to check the validity of the credit card of an individual card holder. We can formulate a goal called *CreditCardCheckGoal* in WSML in order to request suitable Web services to provide this functionality (see Listing 2 in Annex B). The goal is expressed by its request on services which possess the desired capability. The capability is expressed as a post condition that contains a desired outcome of the information space of the Web service after invocation using an axiom which is expressed formally as a conjunction of logical expressions.

In order to find appropriate Semantic Web services that match the requirements of the goal, an infrastructure of backend components is needed to process the goal, match it with existing service capability descriptions, and select appropriate services to compose the required functionalities. Such a semantic infrastructure consisting of the aforementioned SOA4All backend components makes up a runtime semantic system called the semantic execution environment (SEE) (see D6.4.1 and D6.5.1).

6.3 An Architecture for Semantically-Annotated ES

SAP ES expose business functionalities and will be used in WP7 to realize processes such as the registration of a business described in D7.2. The integration of ES into the SOA4All SEE poses several challenges. First, SAWSDL annotations to express WSMO-Lite semantics cannot be applied directly to existing WSDL service descriptions because it is often not desirable from an administrative and security perspective. Furthermore, some ES generate their WSDL interfaces on the fly during service deployment so that it is impractical to append semantic annotations to these dynamically generated WSDL interfaces without modifying the service implementation or the deployment mechanism.

Instead, we introduce an additional *integration layer* as depicted in Figure 12 to allow the semantic annotation of WSDL descriptions with SAWSDL references at the integration layer. With this approach, each WSDL of an existing ES will have a corresponding semantically-annotated WSDL in the integration layer. This maps syntactically to the non-annotated WSDL interface of the corresponding service on the backend enterprise system with the augmentation of the additional SAWSDL annotations so that it identifies the same service contract of the referenced Web service on the backend. The integration layer forwards or relays the SOAP messages dedicated for the ES transparently between the Web service instances and the semantic layer. The semantic annotations for existing SAP ES can be created by using, e.g., the WSMO-Lite editor developed by T2.1 (see D2.1.2).

To fulfill the promises of the functions listed at the beginning of this section, the semantic layer follows software design principles such as the layer approach and separation of concern to maintain a high level of coherence between the components. The integration layer itself is actually a set of Web services that proxy Web service invocations on behalf of the semantic layer without adding extra business logic to the existing ES on the backend.

We propose to implement the necessary integration layer services as Java EE EJB port components, for instance, as stateless Enterprise JavaBeans [EJB2006] that will be deployed as Java Web services. One of the advantages of this approach is to build the indirection using the service reference annotation mechanism into the EJB components in order to let them refer to the existing backend ES. This indirection decouples existing ES from the semantic layer by channeling the service invocations through the integration layer. To the semantic layer and its Web service invocation mechanism, the EJB components of the integration layer appear as semantic Web services with SAWSDL annotations.

Another challenge for handling ES in the context of SOA4All is their large and complex service interfaces and their complex behavior that may lead to unforeseen side-effects (see Figure 9). The consumption of such a service requires detailed application-specific knowledge of the invocation messages, syntax format, and an understanding of the meaning of the message parameters. In order to reduce this complexity (partially), the integration layer also acts as a façade to an ES by intercepting invocation requests from the semantic layer and filling out parameters with appropriate values that can be derived, e.g., from the user context.

From a flexibility point of view, existing ES can be replaced seamlessly. Since we do not implement heavyweight Web services at the integration layer, replacing existing ES with new services will require only minor adaptation at the integration layer. Actually integrating a specific ES requires a customization and redeployment of its pendant Web service in the integration layer that cannot be automated. Furthermore, application-relevant expertise by a service engineer will be needed to provide the semantic annotations while being supported in this task by the SOA4All Studio tools.

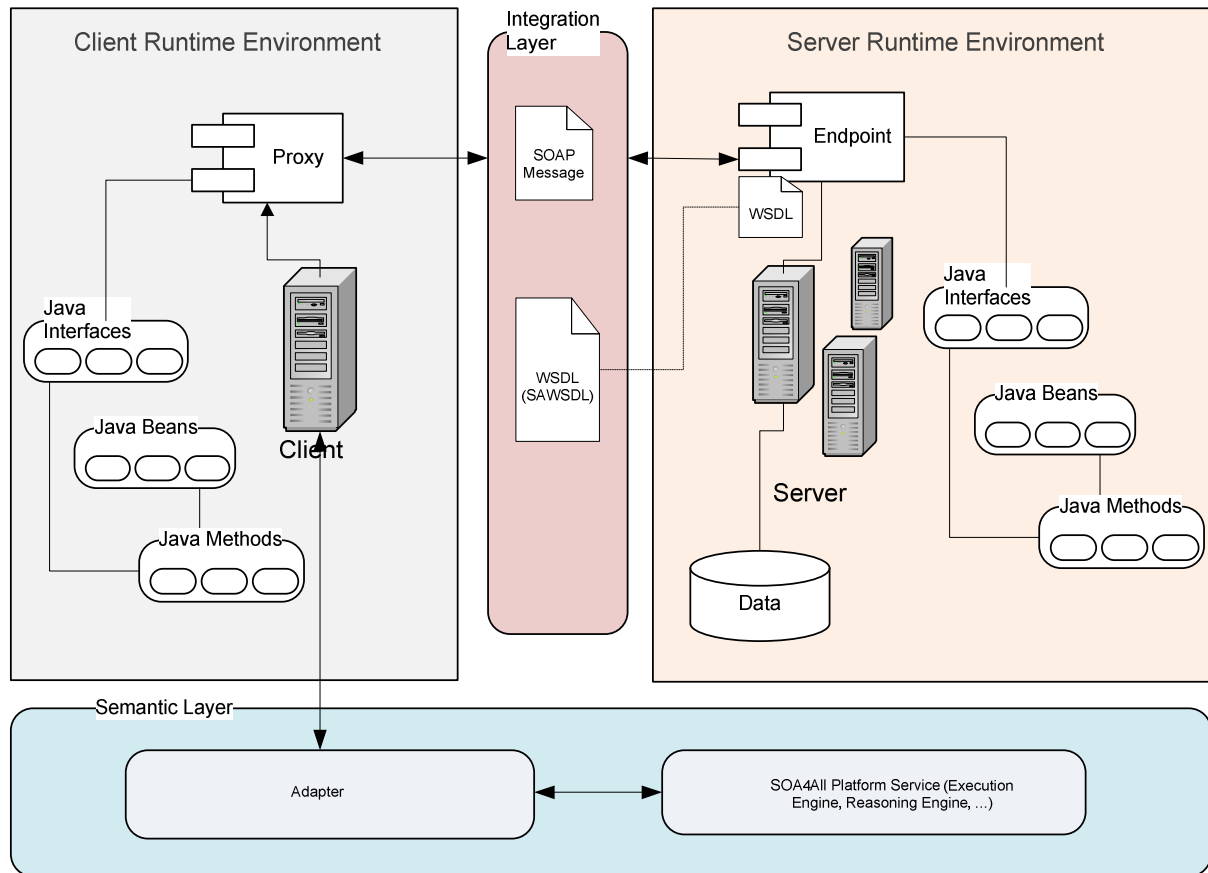


Figure 12: Web Service Architecture with Semantic Layer & Integration Layer

Finally, from the perspective of message exchange, the integration layer can incorporate built-in message handling functionalities where necessary. One typical example of message handling is the management of access authorization for the existing ES using message-based user authentication. Nearly all of the SAP ES are deployed on the SAP NetWeaver platform application server that uses client authentication to restrict access to the ES. The EJB components can be used to transform request messages by appending authentication information to the request messages seamlessly, eliminating the need for the semantic layer to attend to the management of additional authentication information. Traditionally, this authentication information can either be included in a dedicated section of the HTTP or the SOAP header.

Furthermore, in order to deal with the need of other authentication mechanisms such as digital certificates or encrypted tokens, etc. of some other Web service platforms, the integration layer should be able to handle message transformations with regards to application-specific requirements on how to transmit this authentication token. One proven approach for message transformation is to utilize the message handler mechanism on the Web services runtime environment. A message handler is a runtime component located on the message path between the integration layer and the corresponding ES on the backend. It is used to perform well-defined operations on the SOAP messages that pass through it. If the access to a set of ES requires a certain specific type of authentication mechanism, for instance, simple password based authentication, a message handler can be implemented to intercept request messages on the path targeted for the set of ES. The handler will be responsible for transforming the SOAP request messages by including the authentication token in a predefined way.

7. Evaluation Workshop

According to the evaluation plan outlined in Chapter 7 of D7.2, the first stage of end user evaluation will be conducted in the form of a focus group evaluation. The workshop will cover various aspects of software-driven service provisioning and the possible ways in which different services can be composed dynamically to achieve complex tasks. WP7 will organize the workshop at the University of Manchester. The workshop offers an important opportunity for system designers to carry out preliminary studies and to ensure that all end user requirements are collected. At the same time, it offers an opportunity for public sector representatives from the Manchester Town Hall to learn about the envisioned service delivery platform and on how recent developments in service-based technologies can help in providing flexible services.

The discussions in focus group will encourage participants to share and discuss their opinions and experiences related to the SOA4All vision as instantiated by the WP7 scenario. The specific design choices underpinning Web service composition as envisioned by SOA4All will also be discussed. The result of the focus group will be used as input to the software development of the first SOA4All prototype. In summary, the main objectives of this workshop are to:

1. Obtain general opinions of the end users about end user development of service-based software in the public sector domain;
2. Evaluate the current mockups of the composer editor as customized by WP7 within a participatory design process;
3. Capture as many composition editor requirements as possible.

We aim to have at least 15 participants for the planned session, to be divided into 3 groups of 5 participants in each group, plus one moderator. The participants' profiles should match the profiles of the target users envisioned by WP7. The session will last for about 3.5 hours in a large seminar room with 3 round tables seating 6 each, including a 20-minute break in the middle. A 30-minute introductory talk will be followed by a 20-minute discussion on the perceptions about risks and benefits of the envisioned mode of user-driven service composition, and on existing practices and proposed supporting actions (also see Chapter 3 of D7.2). A short notational study will discover how participants understand core proposed representations of the lightweight modeling language (see D6.3.1 and D2.6.1). After the break, the discussion will focus on alternative designs for an end user tool for Web service composition in the public sector domain. Questionnaires and audio tapes will be used to record the participants' responses for later analysis.

Each group will discuss two of three alternative designs. The moderator of the workshop should encourage and facilitate free-flowing discussion by:

1. Going through the design rationale behind each alternative and explain the related mock-ups;
2. Asking the participants to answer questions related to the current alternative by filling in questionnaires;
3. Iteratively repeating Steps 1 and 2 for the other alternative;
4. Discussing the collected answers by all participants.

7.1 Designs for Composing Web Services

The workshop will focus on three alternative approaches for composing Web services and their potential design problems within the SOA4All Composer:

1. data flow to create stateless service mashups
2. control flow using the SOA4All lightweight process modeling language (see D6.3.1)
3. assisted modeling with wizards that guide the user in a stepwise procedure

Each of these design approaches will be covered by one group of participants with the aim of capturing their ideas, feelings, and thoughts. Further details of the design approaches can be found in the generic workshop proposal in D2.5.1.

7.2 Design Rationale

For effective design exploration, we suggest the use of a combination of design rationale, use case scenarios, and - if possible - early prototypes. Design rationale aims to support system designers by representing the argumentation and reasoning behind the design process. This justification of design decisions is important to understand, change, or recreate a design. In this workshop, the Issue-Based Information System (gIBIS), an argumentative notation, will be used to represent issues, arguments, and resolutions. The graphical representations in Figure 13 and Figure 14 show pathways starting from the root node (corresponding to the issue), to children nodes (corresponding to the solutions), and to leaf nodes (corresponding to the arguments). These design rationale diagrams enable comparing and establishing relationships between the various design solutions. For each issue, a gIBIS design rationale is produced as given in Figure 13 and Figure 14.

7.3 Top-level Issues

The workshop will cover several topics related to general end user development in the public sector domain (also see Chapter 3 in D7.2). In this respect, information will be collected about the software tools used by civil servants in their daily jobs. Of particular interest are information about software tools used for service composition in the public sector domain and their frequency of use, if any.

Apart from software usage, an important aspect of this workshop will be to acquire information about the background of end users, their general ideas and thoughts about software-based service composition. This type of information will be acquired by asking specific questions like:

- What information/ system parts do you consider is important for your job?
- What features do you like about your current tools?
- What features do you dislike about your current tools?
- What aspects do you consider problematic in your current tools?

Other questions can be focused on end user development issues in the public sector:

- What are the benefits of end user development?
- What are the risks of end user development?

- What strategies / approaches do end users follow when developing applications?

Apart from covering these general aspects, the workshop will specifically focus on the design choices underpinning the SOA4All Composer (see D2.6.1). In this respect, screenshots or mockups of the currently developed tools will be shown on the wall or distributed in hand-outs. Workshop participants will be asked to provide their feedback on the initial design of the SOA4All Studio tools. The feedback of workshop participants will help in determining whether the currently developed tool (represented by screen shots or mockups) will be able to address the potential problems in their daily jobs.

The detailed list of issues that will be discussed in the workshop is specified in the generic workshop proposal (for WP2, WP7, and WP8) in D2.5.1. The later proposal also contains the workshop agenda that describes the sequence of activities in the proposed workshop.

The feedback and recommendations of workshop participants will be recorded during the workshop. Once the workshop has been carried out, the participants' responses will be collected and analyzed. Recorded material and unstructured questionnaire responses will be analyzed using the Grounded Theory approach, quantitative questionnaire answers will be analyzed using conventional statistical methods.

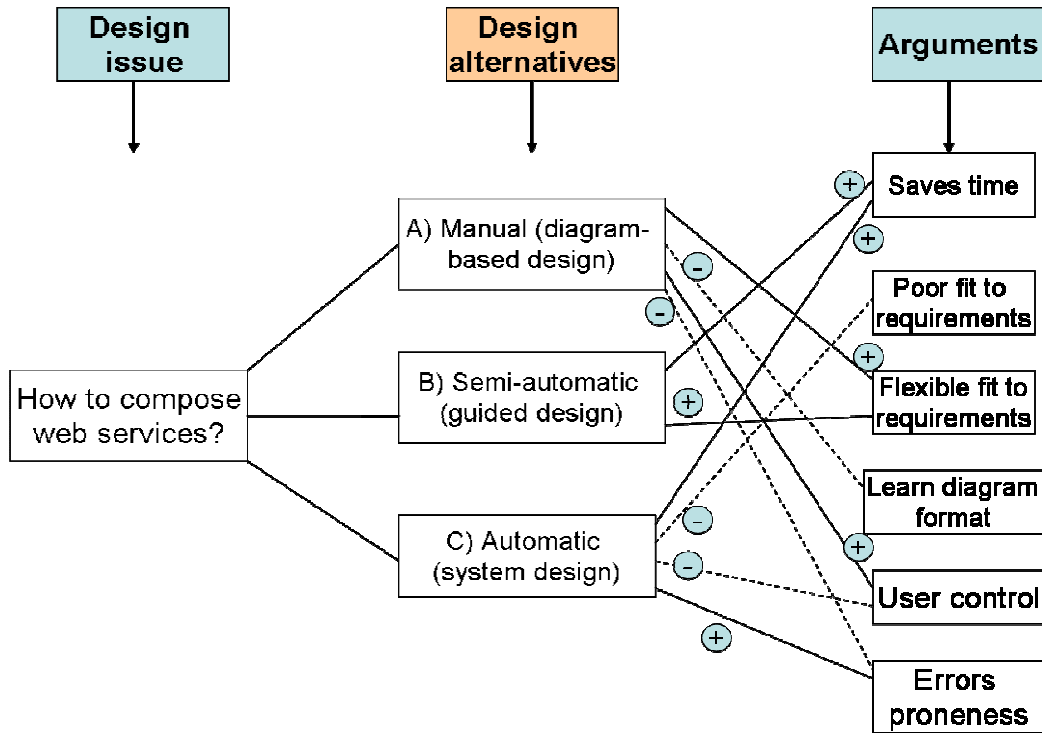


Figure 13: gIBIS Design Rationale for Composition (Issue 1)

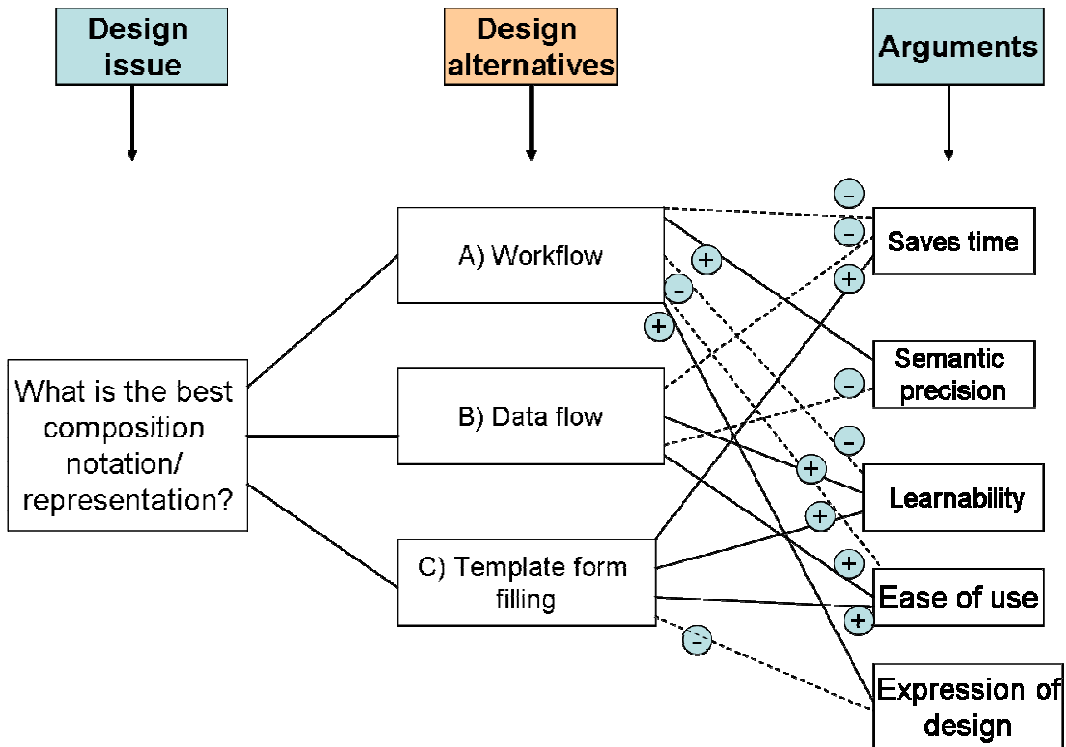


Figure 14: gIBIS Design Rationale for Composition (Issue 2)

8. Conclusions

WP7 will implement an open service delivery platform that allows civil servants to handle typical administrative procedures (such as an application for registering a new business). More specifically, using the Web-based tools of the SOA4All Studio, civil servants can search, model, annotate, modify, share, analyze, and execute administrative procedures in the form of lightweight processes. These processes may be composed of Enterprise Services (hosted by SAP), public Web services (hosted by 3rd party service providers), and human activities (to be executed by end users).

The service delivery platform can largely be implemented by leveraging the functional components provided by the technical work packages of SOA4All. But in order to meet all requirements of the use case scenario, some customizations and extensions will be developed within WP7. Thus, in addition to investigating the public sector business case and to providing a basis for the technical validation and end user evaluation of the project results, the key technical contributions of WP7 to SOA4All are an adaptation and integration layer so that SAP Enterprise Services can be consumed in SOA4All, additional tools and services for handling human tasks, and an extended user role model.

The next step in this WP will be to implement a first prototype until August 2009 (M18) that will be based on the SOA4All components from the technical WPs and that will be documented in D7.4. In parallel, the first evaluation workshop will take place with results also reported in M18.

9. References

- [Alur2003] Alur, D., Crupi, J., Malks, D. Core J2EE Patterns, 2nd edition. Prentice Hall PTR, June 2003
- [Bieberstein2005] Bieberstein, N., Bose, S., Walker, L. and Lynch, A., Impact of Service-Oriented Architecture on Enterprise Systems, Organizational Structures, and Individuals, IBM Systems Journal, 44 (4), pp. 691-708., 2005
- [BPEL4People2005] WS-BPEL Extension for People – BPEL4People, A Joint White Paper by IBM and SAP, July 2005, available at http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/BPEL4People_white_paper.pdf
- [Dean2004] OWL Web Ontology Language Reference. W3C Recommendation February 2004. Available at <http://www.w3.org/TR/owl-ref/>.
- [EC2006] Directive 2006/123/EC of the European Parliament and of the Council of 12 December 2006 on Services in the Internal Market, OJ L376 of 27.12.2006
- [EJB2006] Enterprise JavaBeans 3.0 Specification. Sun Microsystems Inc. 2006, Java Community Process, JSR220, Available at <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>, last accessed on 20/03/2009
- [Endrei2004] Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P. I., Luo, M. and Newling, T., Patterns: Service-Oriented Architecture and Web Services, IBM Redbooks, 2004
- [Gartner2007] The Real Future of E-Government: From Joined-Up to Mashed-Up, Gartner Report, Nov 2007
- [König2004] König, P., Enterprise Services Architecture (ESA), HBI-Konferenz: iBonD - intelligent Business on Demand, Munich 2004
- [McBride2004] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. February 2004. Available at <http://www.w3.org/TR/rdf-schema/>.
- [Polleres2009] Polleres, A., Krennwallner, T. et al., XSPARQL Language Specification. DERI 2009, Available at <http://xsparql.deri.org/spec/>, last accessed on 20/03/2009
- [RDF2004] Resource Description Framework (RDF): concept and abstract syntax. W3C Recommendation. February 2004. Available at <http://www.w3.org/TR/rdf-concepts/>, last accessed on 20/03/2009
- [Roman2007] Web Service Modeling Ontology (WSMO). Working draft, February 2007. Available at <http://www.wsmo.org/TR/d2/v1.4/>, last accessed on 19/03/2009

- [SAP2007] SAP AG, Enterprise SOA in a Nutshell, 2007
- [SAP2008] SAP AG, ES Workplace Handbook, 2008
- [SAP2009] SAP AG, Records and Document Management, Enterprise Services Wiki, SAP Community Network Wiki, <https://www.sdn.sap.com/irj/scn/wiki?path=/display/ESpackages/Records+and+Document+Management>, last accessed on 19/03/2009
- [SAWSDL2007] Semantic Annotations for WSDL and XML Schema. Recommendation, W3C, August 2007. Available at <http://www.w3.org/TR/sawSDL>, last accessed on 20/03/2009
- [SOAP2007] SOAP Version 1.2, Recommendation, W3C, April 2007, Available at <http://www.w3.org/TR/soap/>, last accessed on 20/03/2009
- [Steinmetz2008] Web Service Modeling Language – language reference, Final draft. August 2008. Available at <http://www.wsmo.org/TR/d16/d16.1/v1.0/>, last accessed on 20/03/2009
- [Swain1995] Swain J. W, White J. D. and Hubbert E. D., Issues in Public Management Information Systems, The American Review of Public Administration, 1995
- [UN/CEFACT2007] United Nations Centre for Trade Facilitation and Electronic Business, Core Components Technical Specification Version 3.0, Available at <http://xml.coverpages.org/CEFACT-CCTSv30-PR2.pdf>, last accessed on 19/03/2009
- [Vitvar2009] Vitvar, T., Kopecky, J. and Fensel, D., WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web. Working draft. March 2008. Available at <http://www.wsmo.org/TR/d11/v0.2/>, last accessed on 19/03/2009
- [Webservicex.net2009] webserviceX.NET, last accessed on 20/03/2009
- [WSDL2001] Web Services Description Language (WSDL): version 1.1, W3C Note, March 2001. Available at <http://www.w3.org/TR/wsdl>, last accessed on 19/03/2009
- [WSDL2007] Web Services Description Language (WSDL): version 2.0. Recommendation, W3C, June 2007. Available at <http://www.w3.org/TR/wsdl20/>, last accessed on 19/03/2009
- [WSML2008] Web Service Modeling Language – language reference, Final draft. August 2008. Available at <http://www.wsmo.org/TR/d2/v1.4/>, last accessed on 20/03/2009
- [WSMO2007] Web Service Modeling Ontology (WSMO). Working draft, February 2007. Available at <http://www.wsmo.org/TR/d2/v1.4/>, last accessed on 19/03/2009

Annex A: Updated Scenario “Registration of a Business”

In this Annex, we update the scenario description for creating, modifying, and executing the administrative procedure to register a business at the City of X. In Chapter 6 of the Deliverable D7.2, the scenario, the users and their roles, and the process model of the administrative procedure itself are discussed in detail. Here, the initial user interface mockups of the scenario are updated to match the current SOA4All Studio design as defined in D2.4.1.

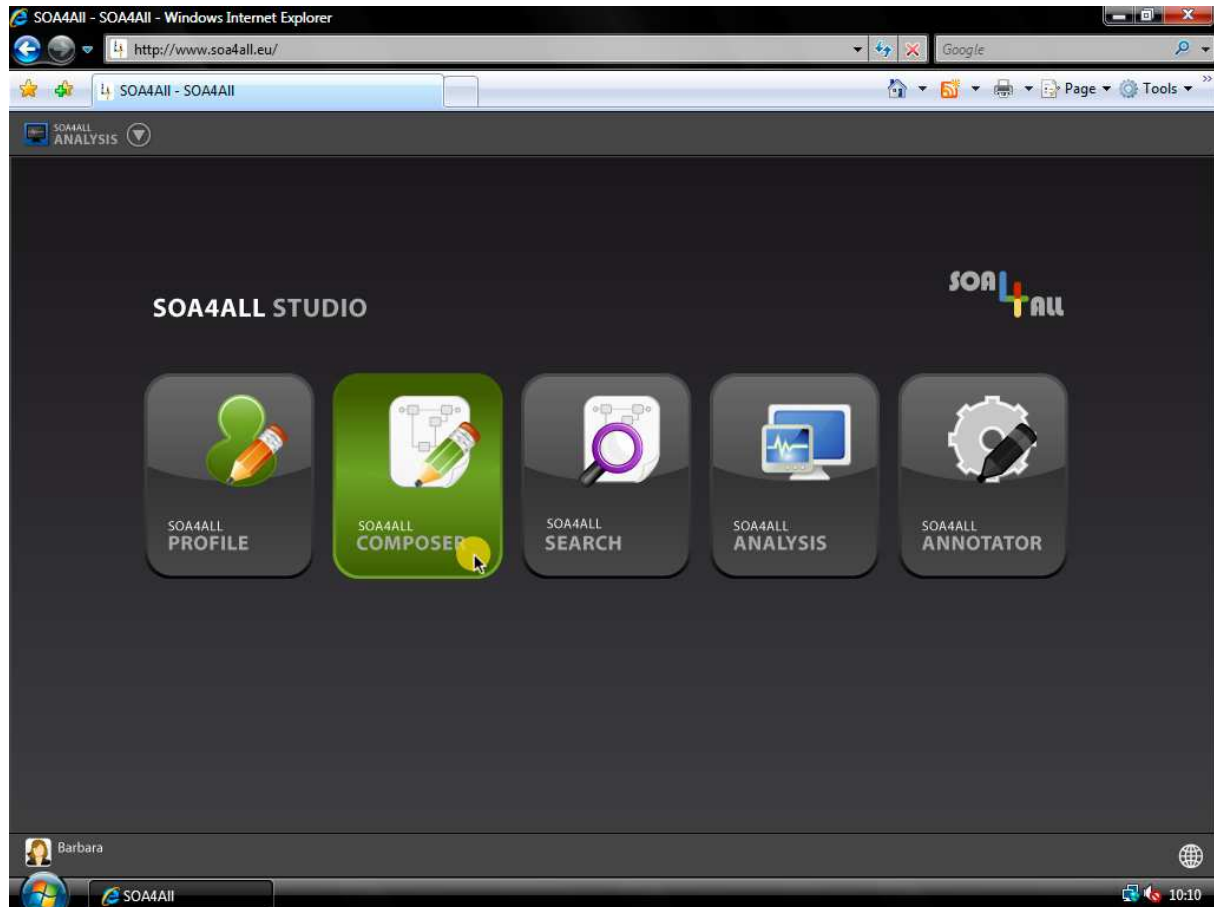
Step 1a

Actor: Barbara

SOA4All Components

WP2 T2.4 Studio and Dashboard

GUI Mockup



Description

Barbara models the first version of the “Registration of a Business” process as depicted in Figure 16, Deliverable D7.2. She logs in to the SOA4All service platform (i.e., the SOA4All Studio) with her browser and starts the SOA4All Composer by selecting the corresponding icon in the SOA4All Dashboard.

Step 1b

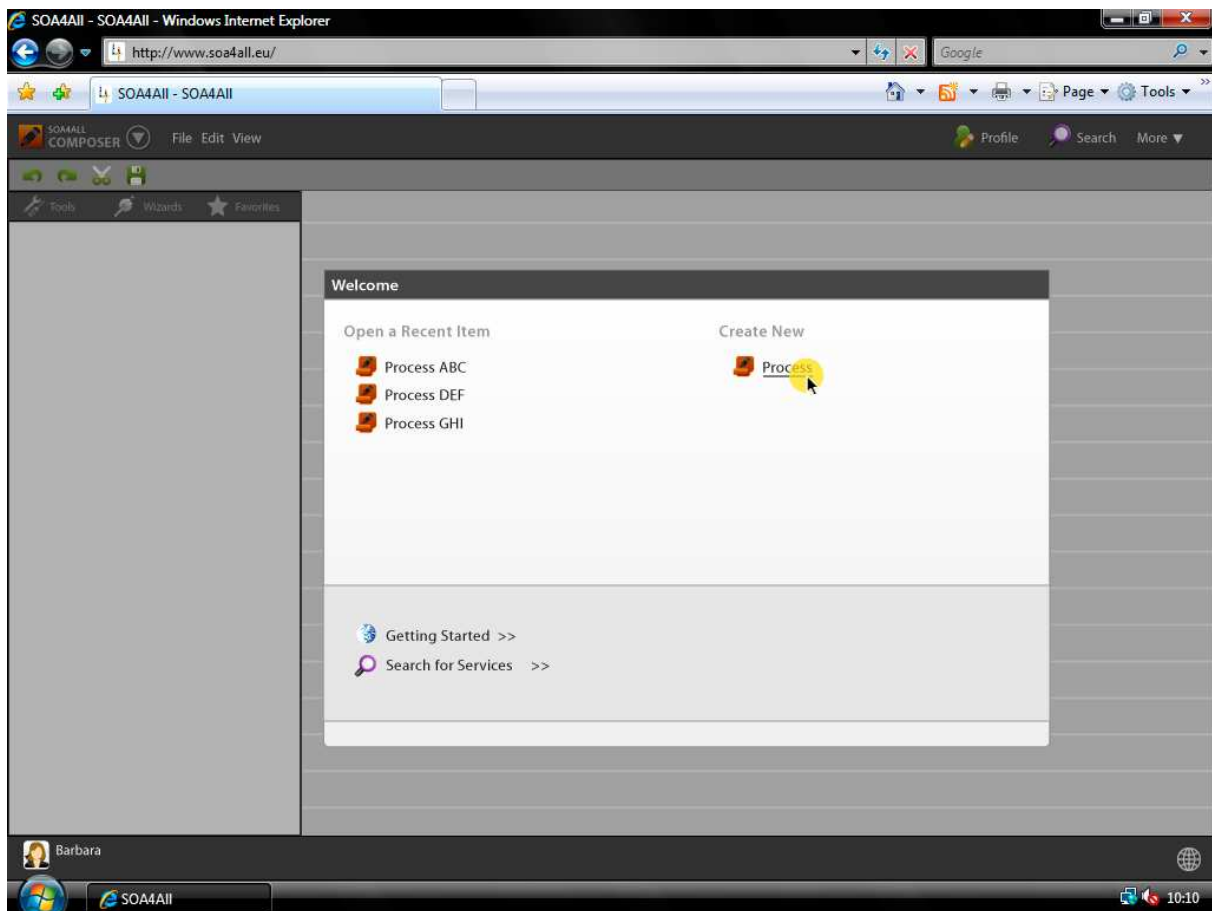
Actor: Barbara

SOA4All Components

WP2 T2.4 Studio

T2.6 Process Editor

GUI Mockup



Description

Barbara draws a new process model for the “Registration of a Business” Process V1: First, she creates a new process.

Step 1c

Actor: Barbara

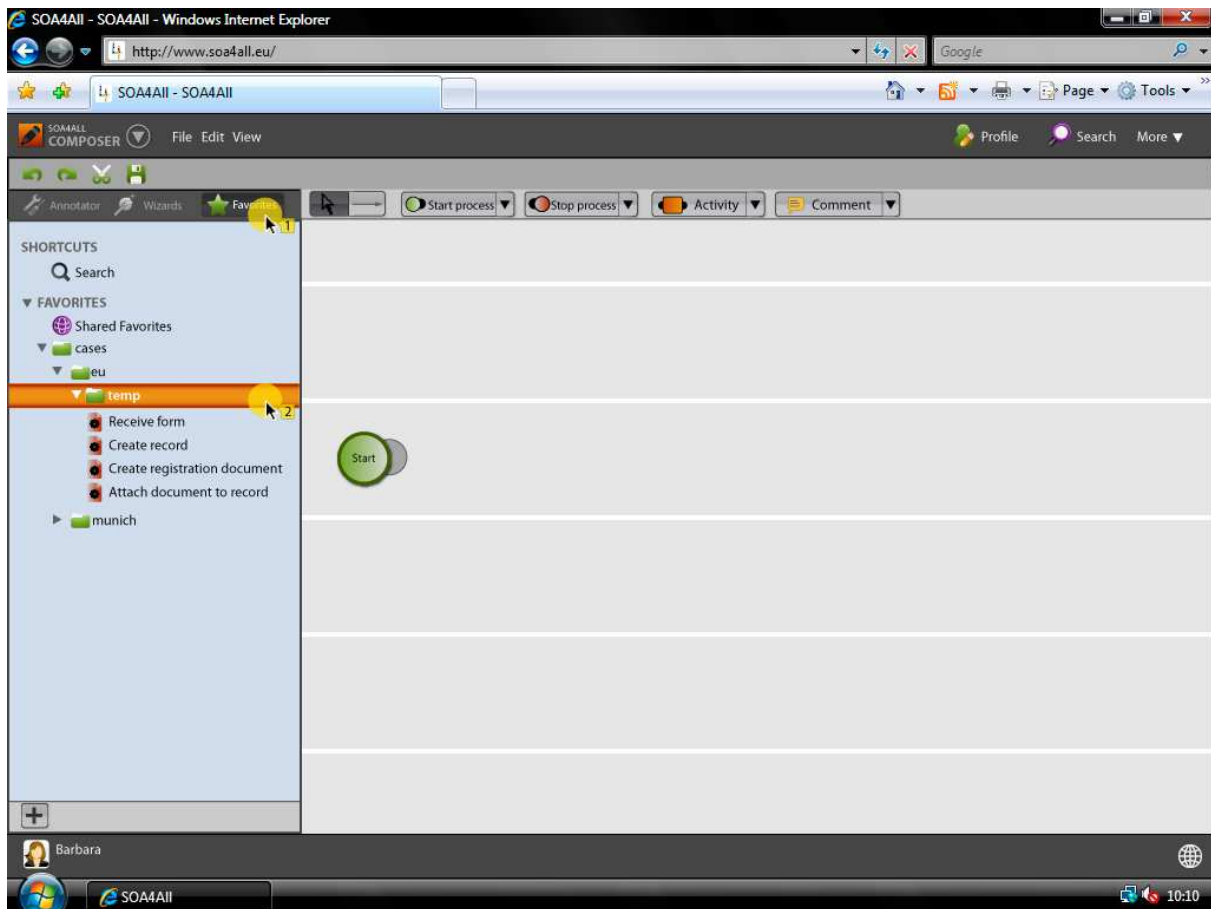
SOA4All Components

WP2 T2.4 Studio

T2.6 Process Editor

WP6 T6.3 Modeling Language

GUI Mockup



Description

From her favorites list containing the services she most frequently uses, Barbara adds the first activity (consisting of the service “receive from”) to the new process.

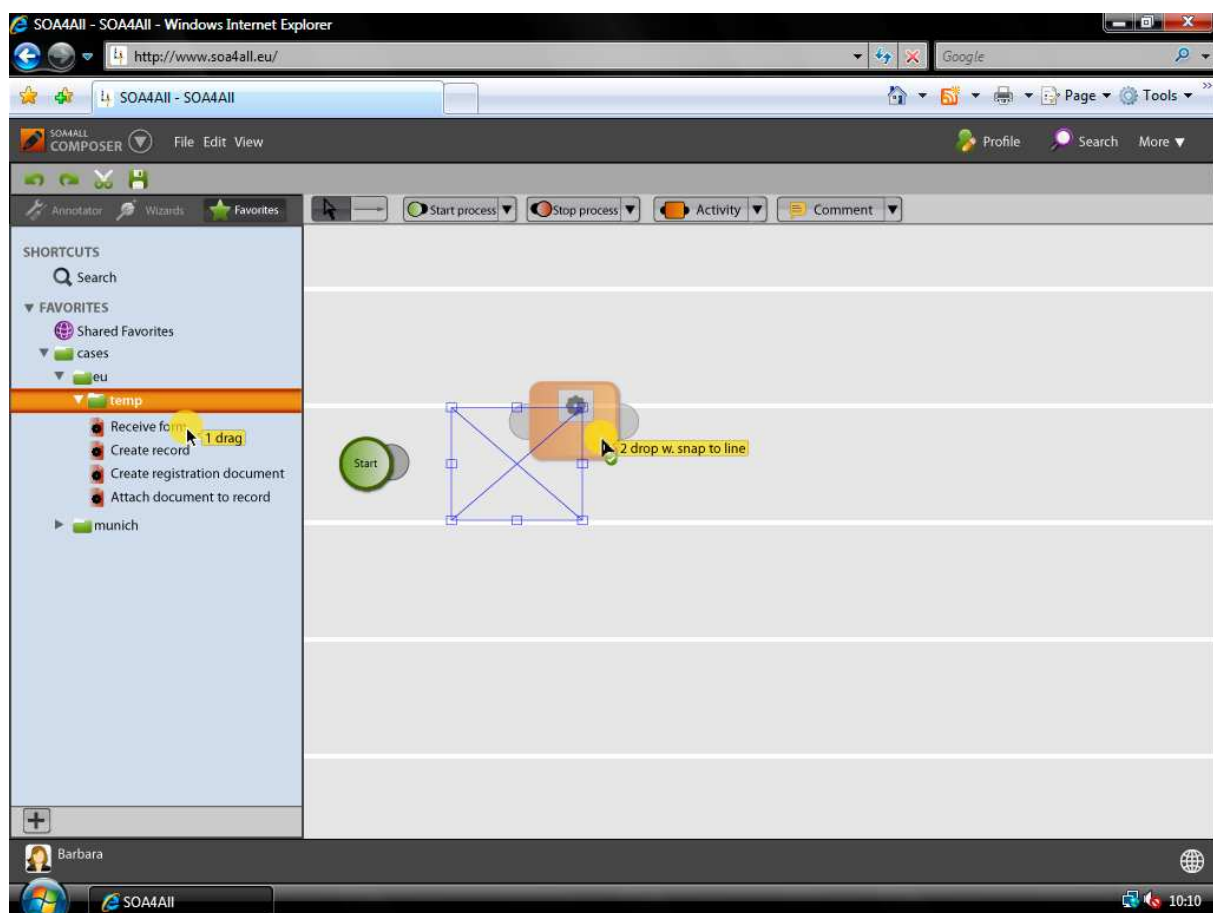
Step 1d

Actor: Barbara

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor
- WP6 T6.3 Modeling Language
- WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

The SOA4All Composer supports her with a “snap to grid” function in her drag and drop operation to create an esthetic layout. The modeling elements she can use for her task are defined by the lightweight process modeling language in T6.3.

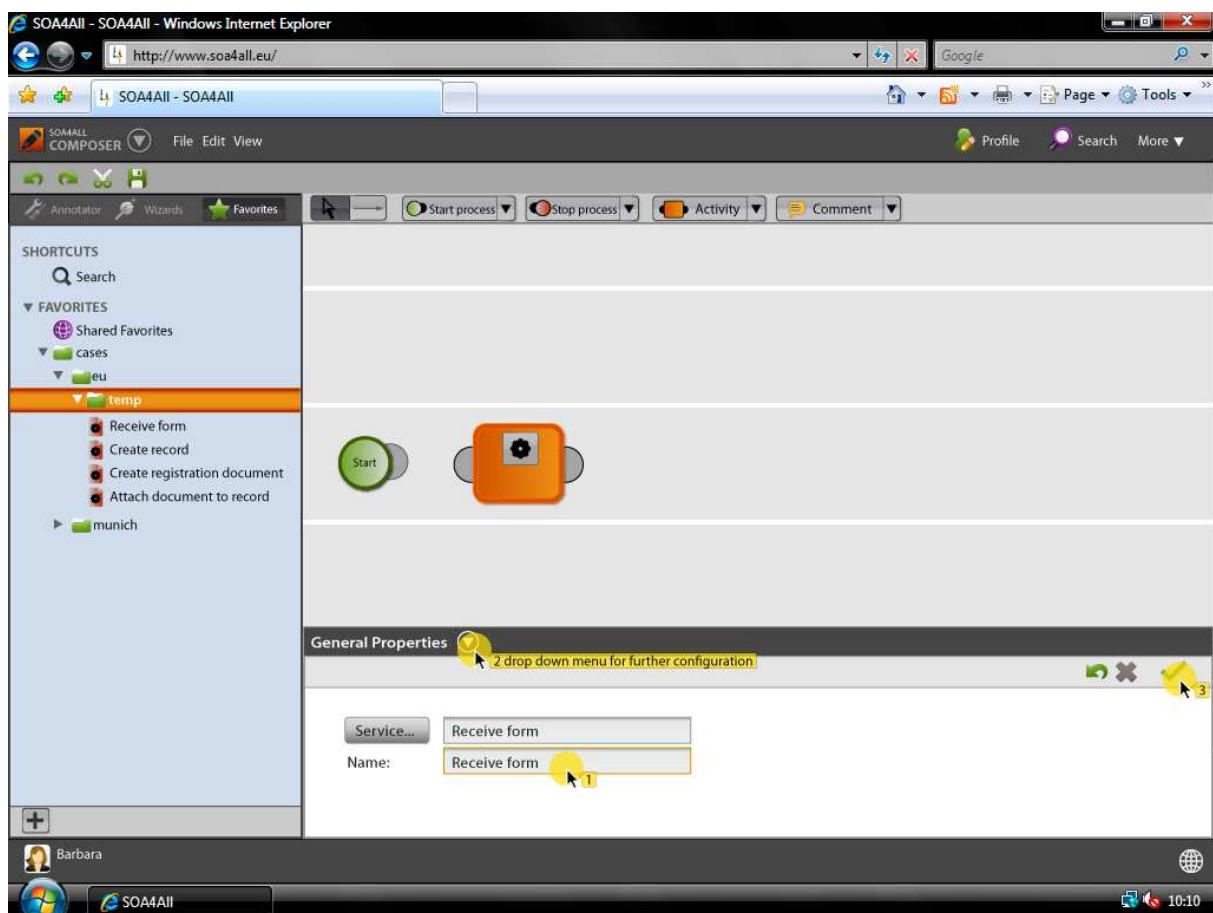
Step 1e

Actor: Barbara

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor
- WP6 T6.3 Modeling Language
- WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

Now Barbara defines the specifics of the concrete service that is called at this step.

Step 1f

Actor: Barbara

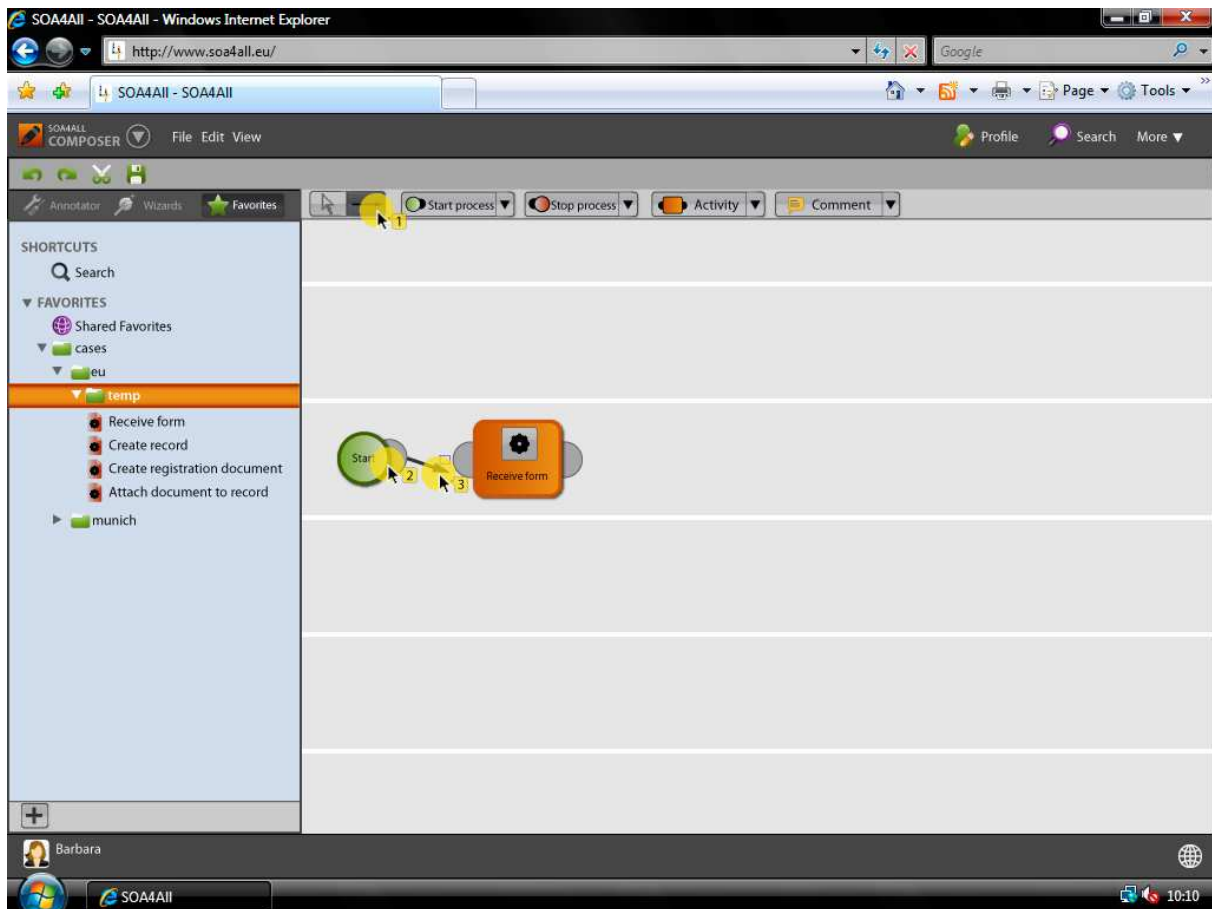
SOA4All Components

WP2 T2.4 Studio

T2.6 Process Editor

WP6 T6.3 Modeling Language

GUI Mockup



Description

Barbara specifies the control flow between the “Start” and the “Receive from” activity by drawing an unconditioned connector between the two.

Step 1g

Actor: Barbara

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor
- WP6 T6.3 Modeling Language
- WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

Repeating Steps (1d), (1e), and (1f), Barbara continues to add activities (that are based on SAP ES in this example).

Step 2a

Actor: Barbara

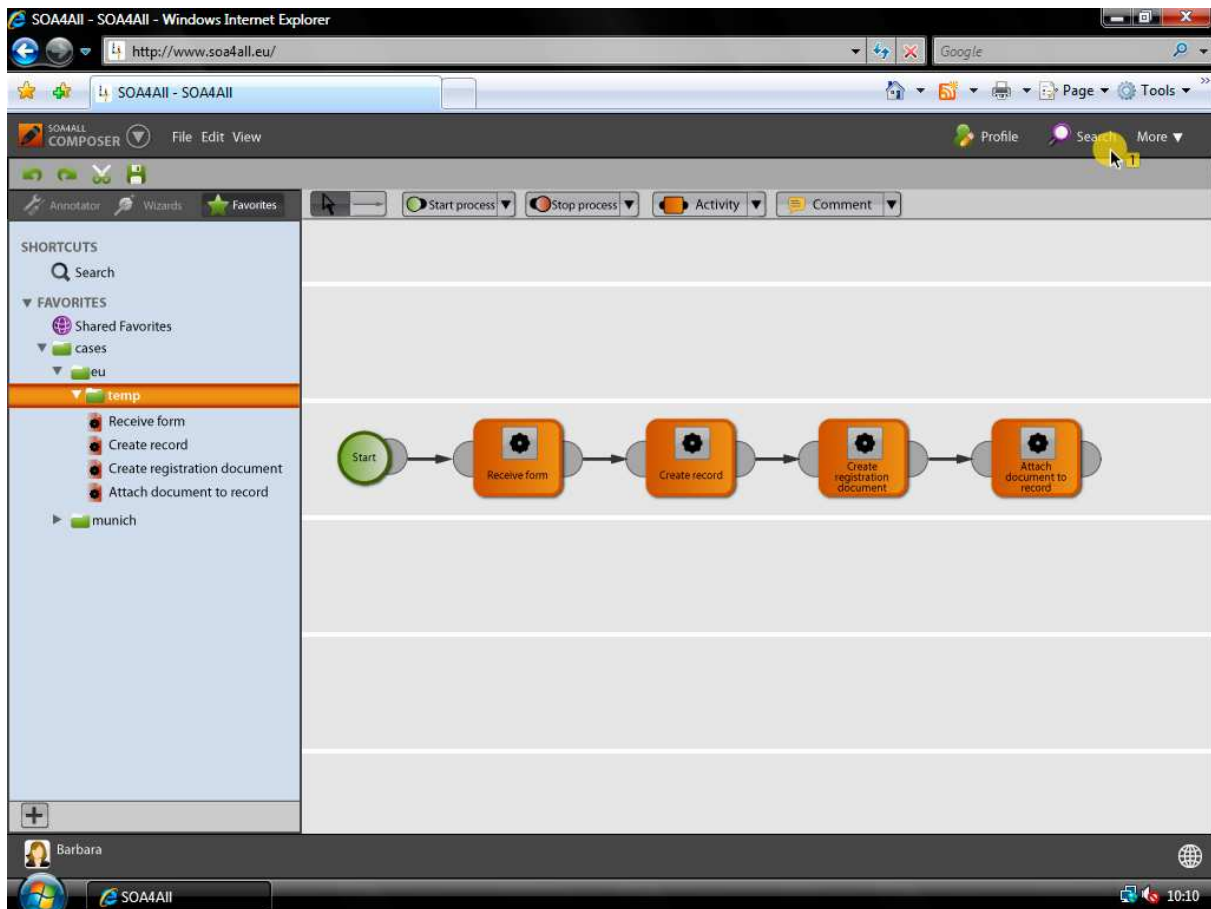
SOA4All Components

WP2 T2.4 Studio

T2.6 Process Editor

WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

Barbara now needs to attach a service, which is not in her Favorites' list yet. Therefore, she opens the SOA4All search interface.

Step 2b

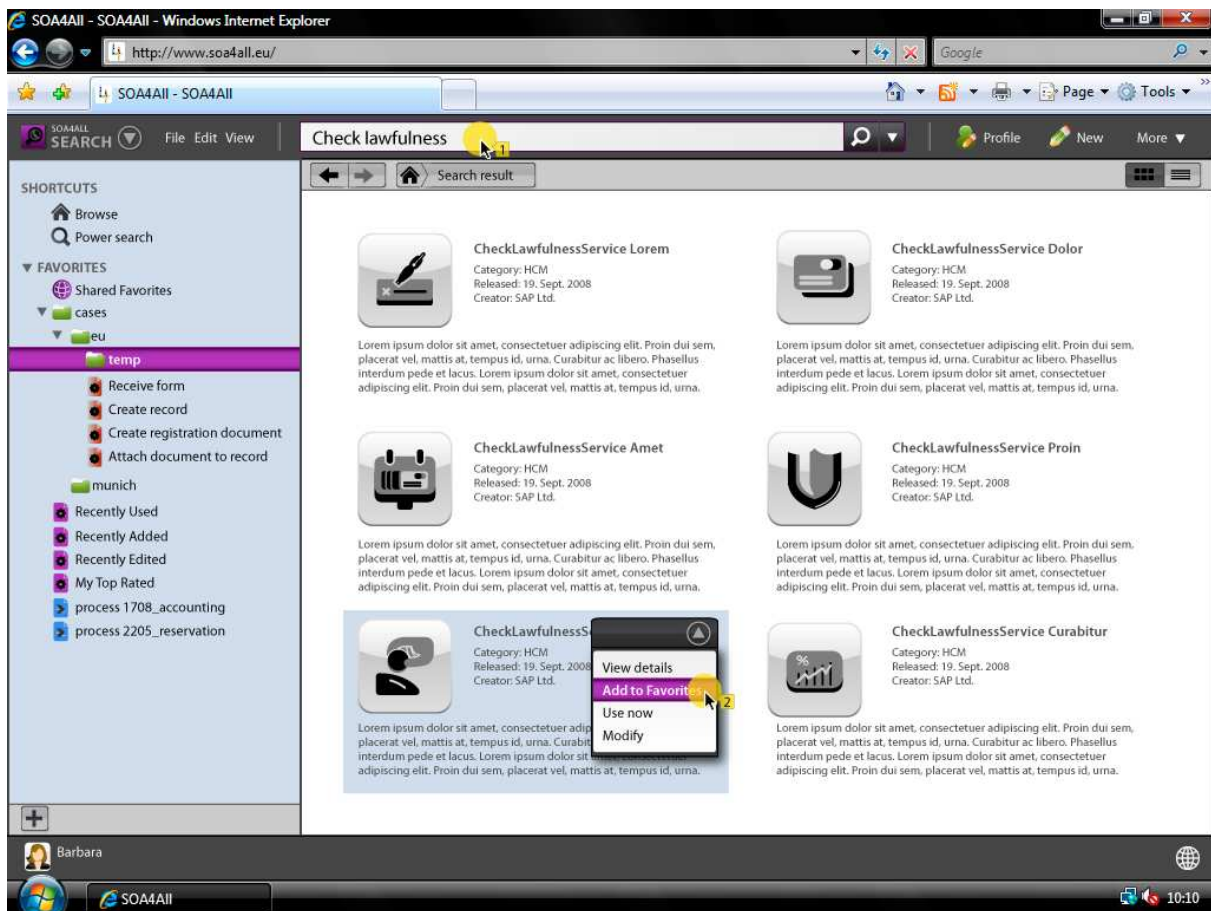
Actor: Barbara

SOA4All Components

WP2 T2.4 Studio, Search Interface

WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

Typing in the right keywords into the textual search interface, Barbara quickly discovers the service she was looking for, bookmarks it in her Favorites' list, and switches back to the SOA4All Composer.

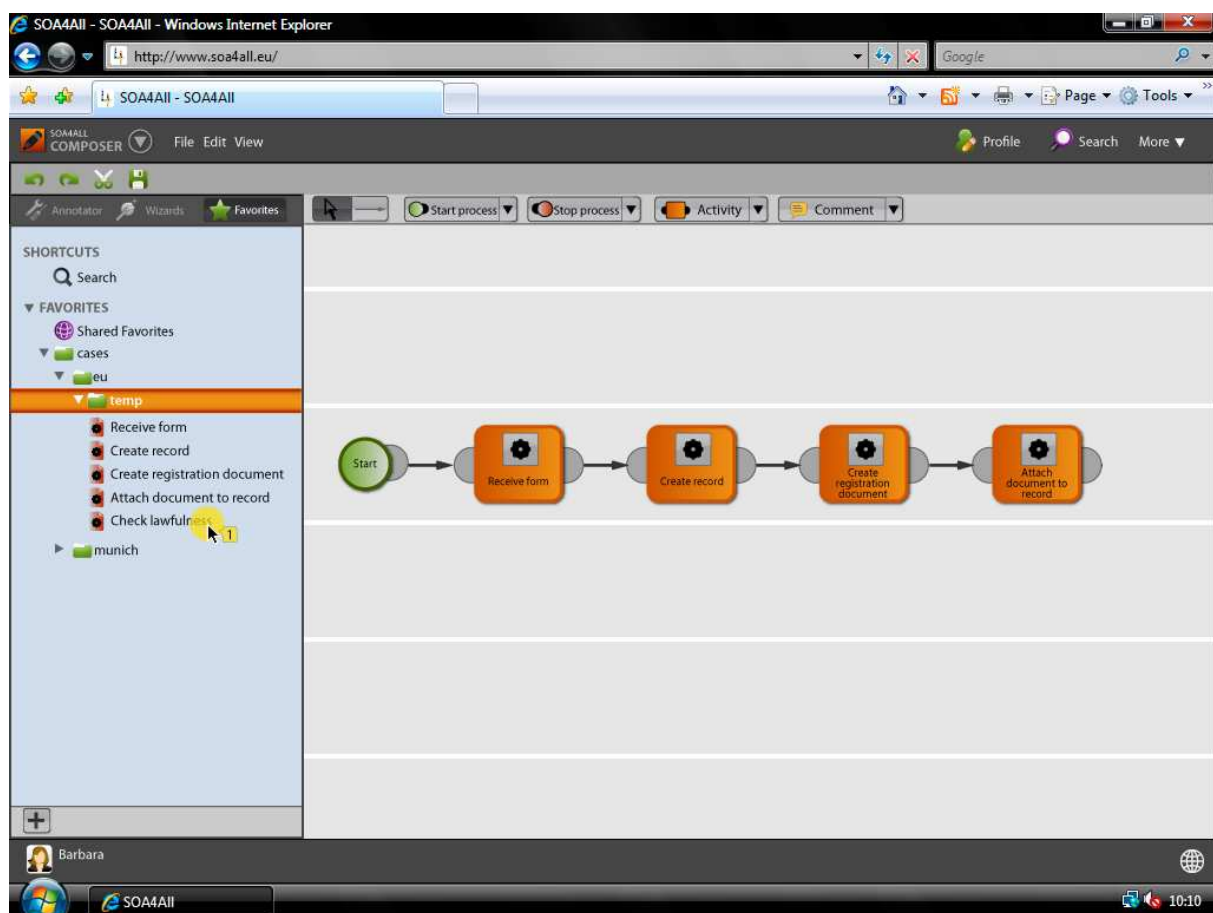
Step 2c

Actor: Barbara

SOA4All Components

- WP2 T2.4 Studio, Search Interface
- T2.6 Process Editor
- WP6 T6.3 Modeling Language
- WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

From her Favorites' list, Barbara can now add the found service to the process as a new activity.

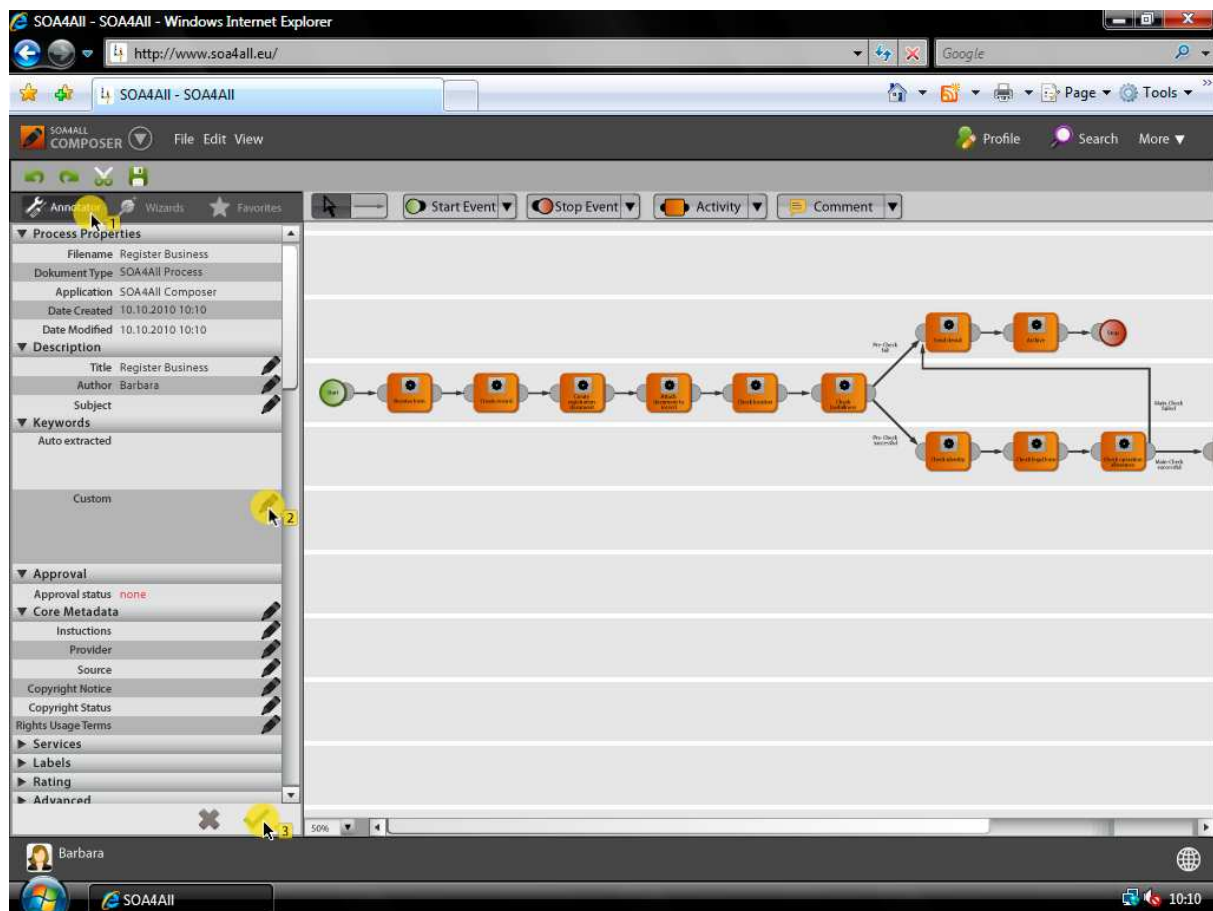
Step 3

Actor: Barbara

SOA4All Components

- WP2 T2.1 Service Provisioning, Service Annotations
- T2.4 Studio
- T2.6 Process Editor
- WP3 T3.4 Semantic Service Description
- WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

By repeating the steps described above, Barbara completes the process “Registration of a Business” V1 as depicted in Figure 16, Deliverable D7.2. Next, she provides the required (semantic) annotations: she selects keywords and categories to be assigned with the process, creates a textual description to be read by her colleagues etc. By default, the process is annotated a having not been checked for compliance yet.

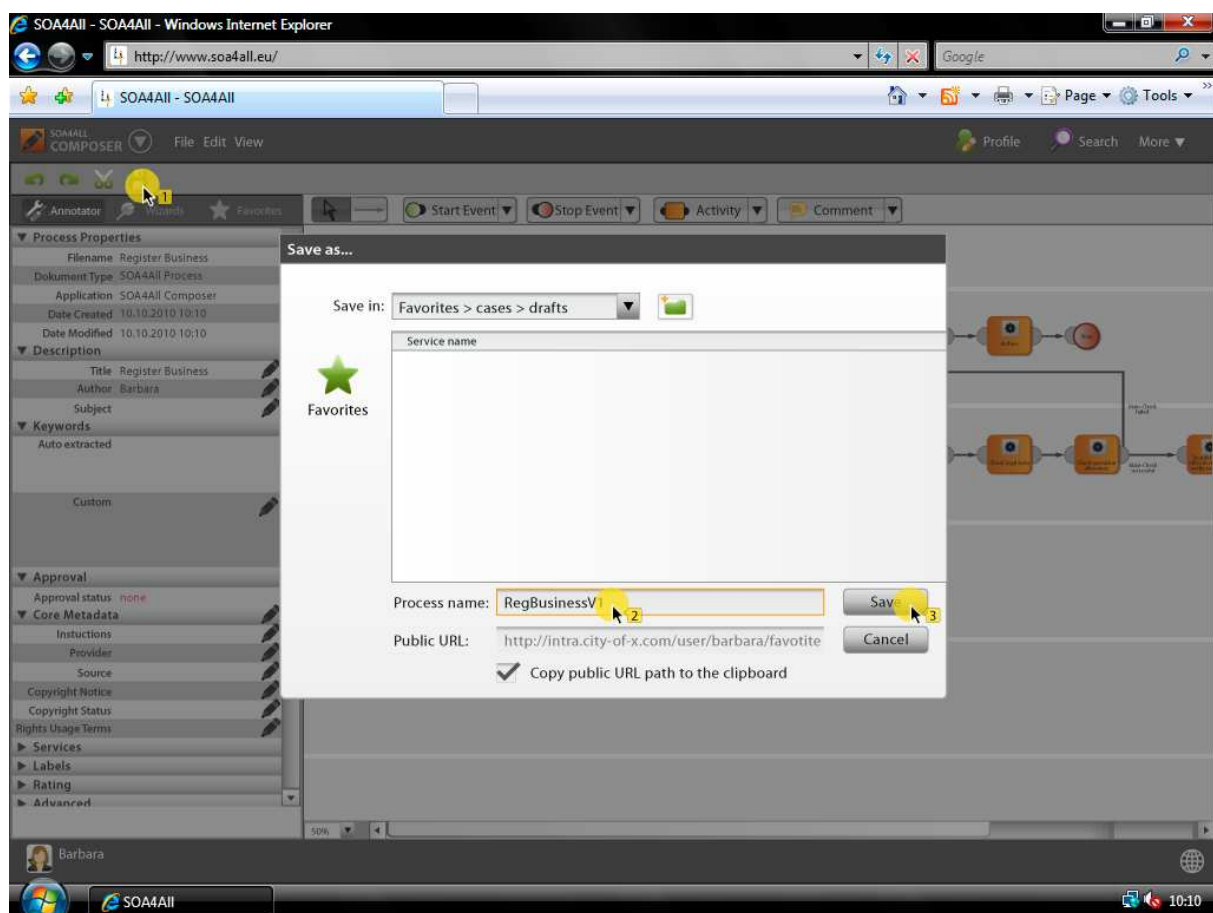
Step 4

Actor: Barbara

SOA4All Components

- WP1 T1.3 Storage Infrastructure
- WP2 T2.4 Studio, Storage Services
- T2.6 Process Editor
- WP6 T6.3 Modeling Language

GUI Mockup



Description

Finally, Barbara saves the process under its name. A unique URI is created to reference the process, and the process becomes available in the shared repository of the City of X so that it can be accessed by her colleagues.

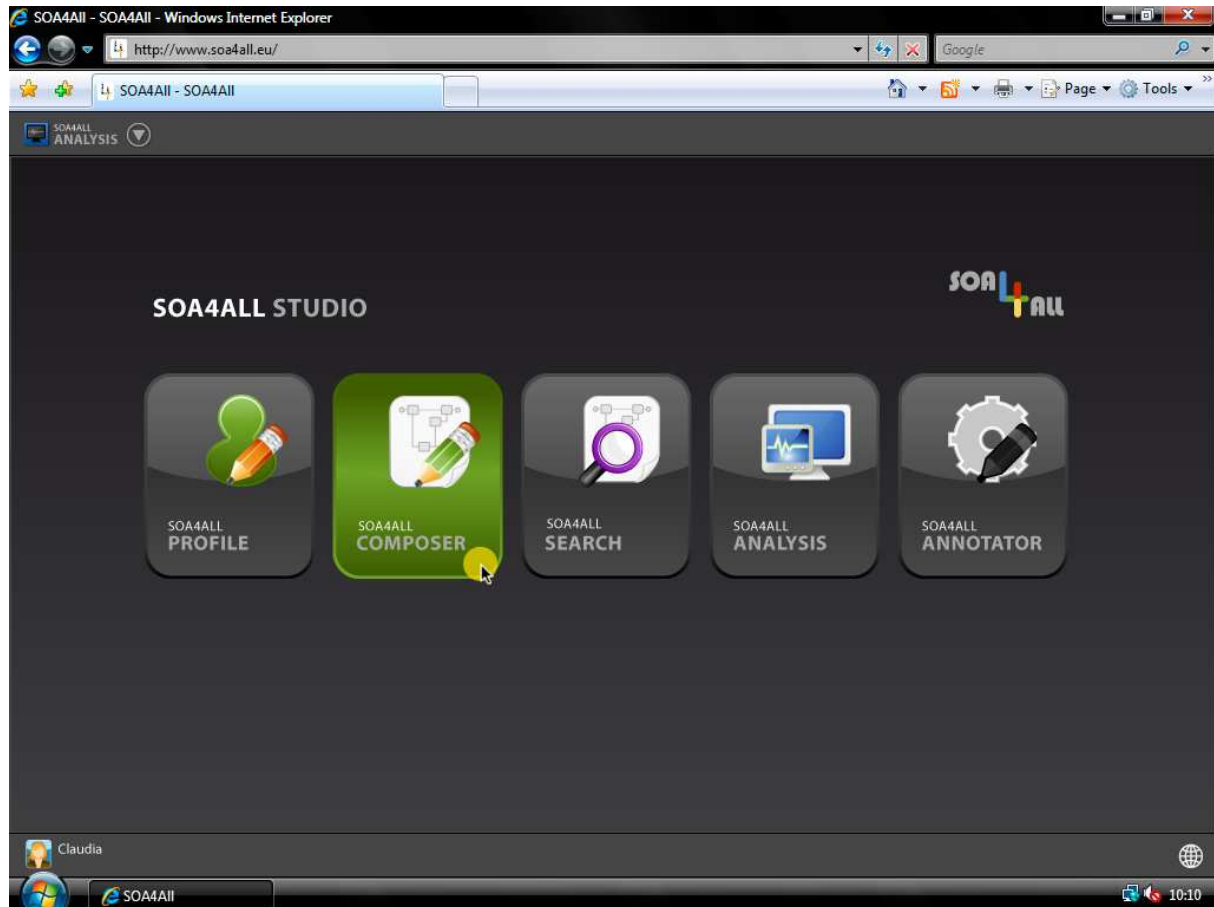
Step 5a

Actor: Claudia

SOA4All Components

WP2 T2.4 Studio and Dashboard

GUI Mockup



Description

Claudia has the task to check the legal compliance of new or updated processes before they can be deployed. She logs in to the SOA4All service platform (i.e., the SOA4All Studio) with her browser. First, she starts the SOA4All Composer by selecting the corresponding icon in the SOA4All Dashboard.

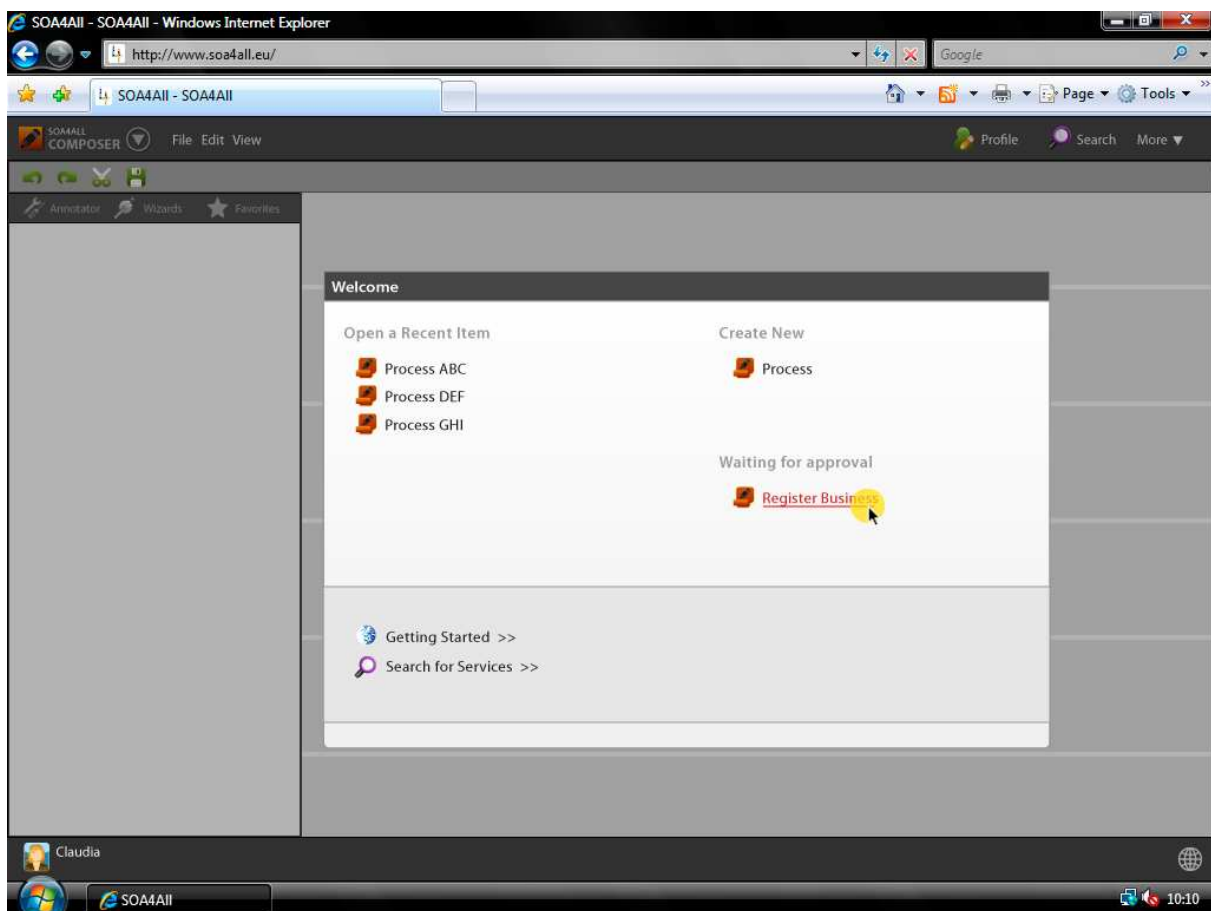
Step 5b

Actor: Claudia

SOA4All Components

- WP1 T1.3 Storage Infrastructure
- WP2 T2.4 Studio, Storage Services, Search
- T2.6 Process Editor
- WP7 T7.4 Customized Prototype

GUI Mockup



Description

After starting the SOA4All Composer, Claudia can see that there is a new process waiting for her approval. The list of such processes can be created automatically by searching for processes with the appropriate annotation.

Step 5c

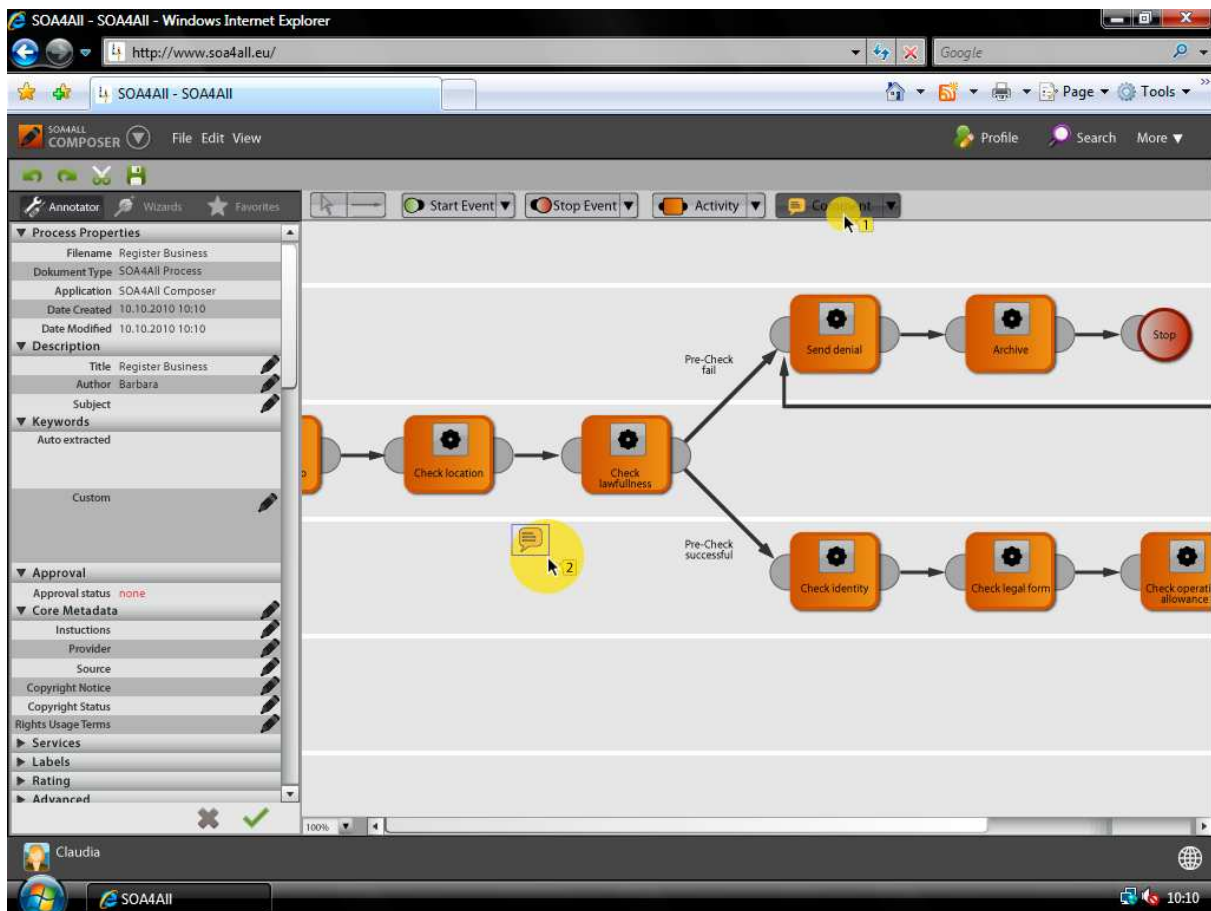
Actor: Claudia

SOA4All Components

WP2 T2.4 Studio

T2.6 Process Editor

GUI Mockup



Description

Claudia reviews the process details in the SOA4All Composer. Satisfied with the compliance of the process, she leaves a textual comment for her colleagues.

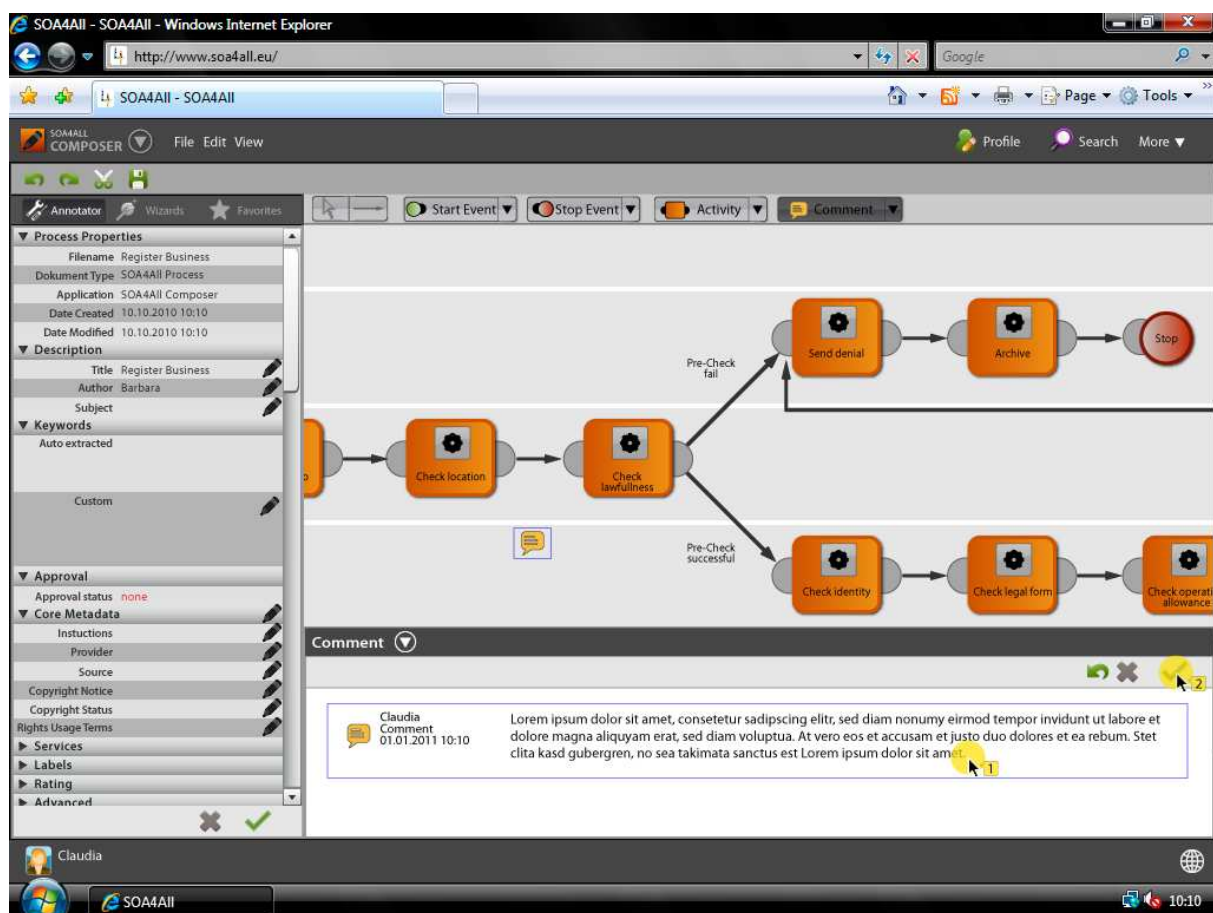
Step 5d

Actor: Claudia

SOA4All Components

- WP1 T1.3 Storage Infrastructure
- WP2 T2.1 Service Provisioning, Service Annotations
- T2.4 Studio, Storage Services
- T2.6 Process Editor

GUI Mockup



Description

Claudia writes her comment, which is attached to the process.

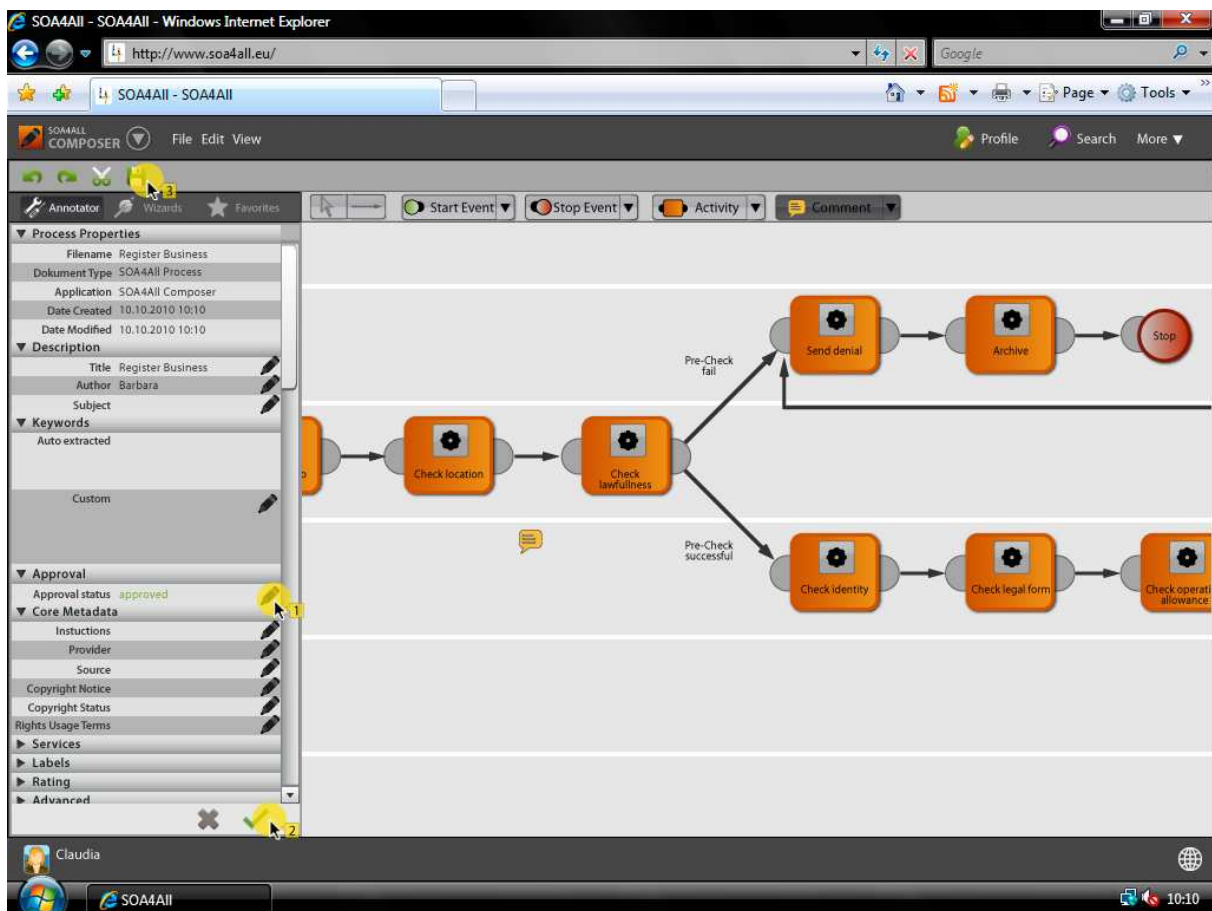
Step 5e

Actor: Claudia

SOA4All Components

- WP1 T1.3 Storage Infrastructure
- WP2 T2.1 Service Provisioning, Service Annotations
T2.4 Studio, Storage Services, Role Model
T2.6 Process Editor
- WP7 T7.4 Customized Prototype

GUI Mockup



Description

Finally, Claudia marks the process as compliant using the service annotation tool.

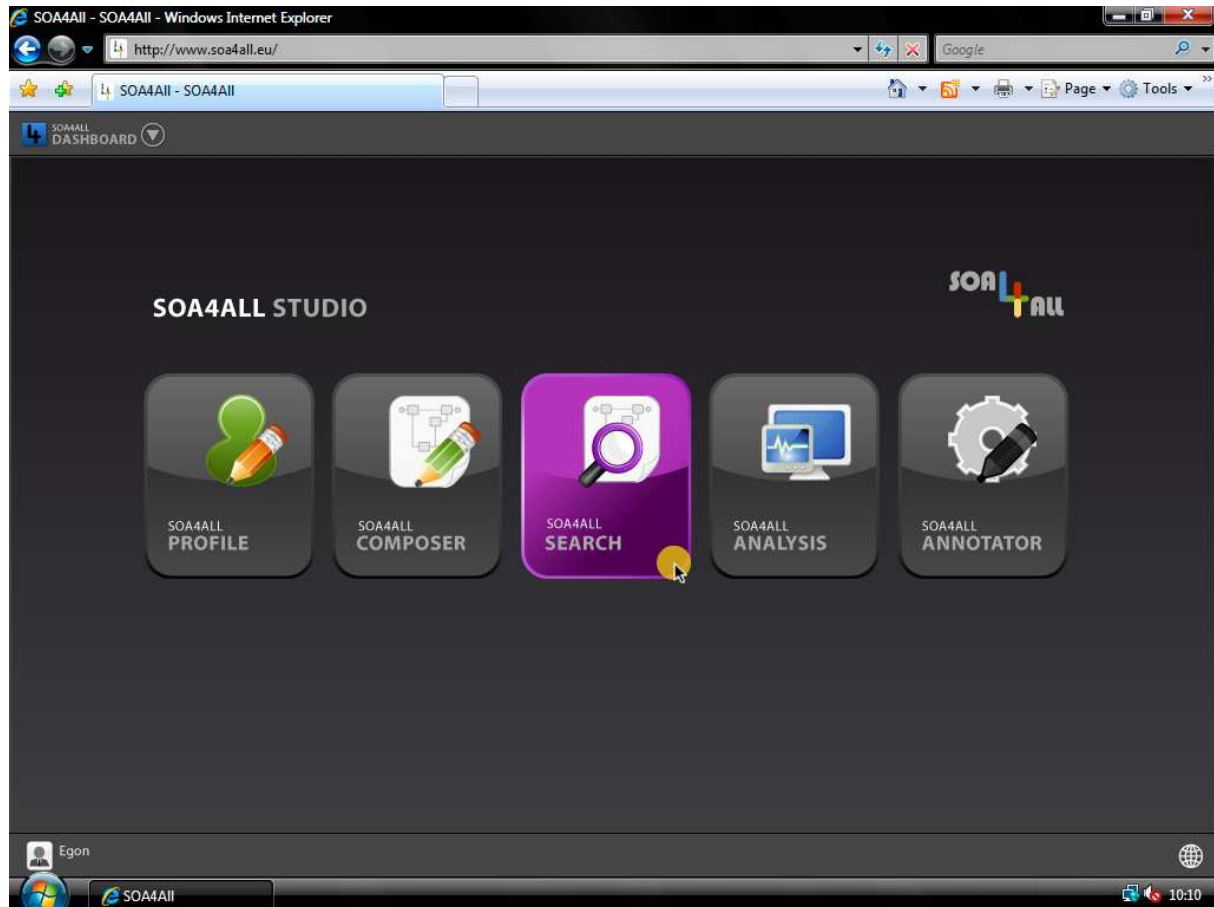
Step 6a

Actor: Egon

SOA4All Components

WP2 T2.4 Studio and Dashboard

GUI Mockup



Description

Egon has the task to adjust the payment part of the “Registration of a Business” process model to include credit card payments. First, he logs into the SOA4All Studio and opens the search tool to retrieve the process.

Step 6b

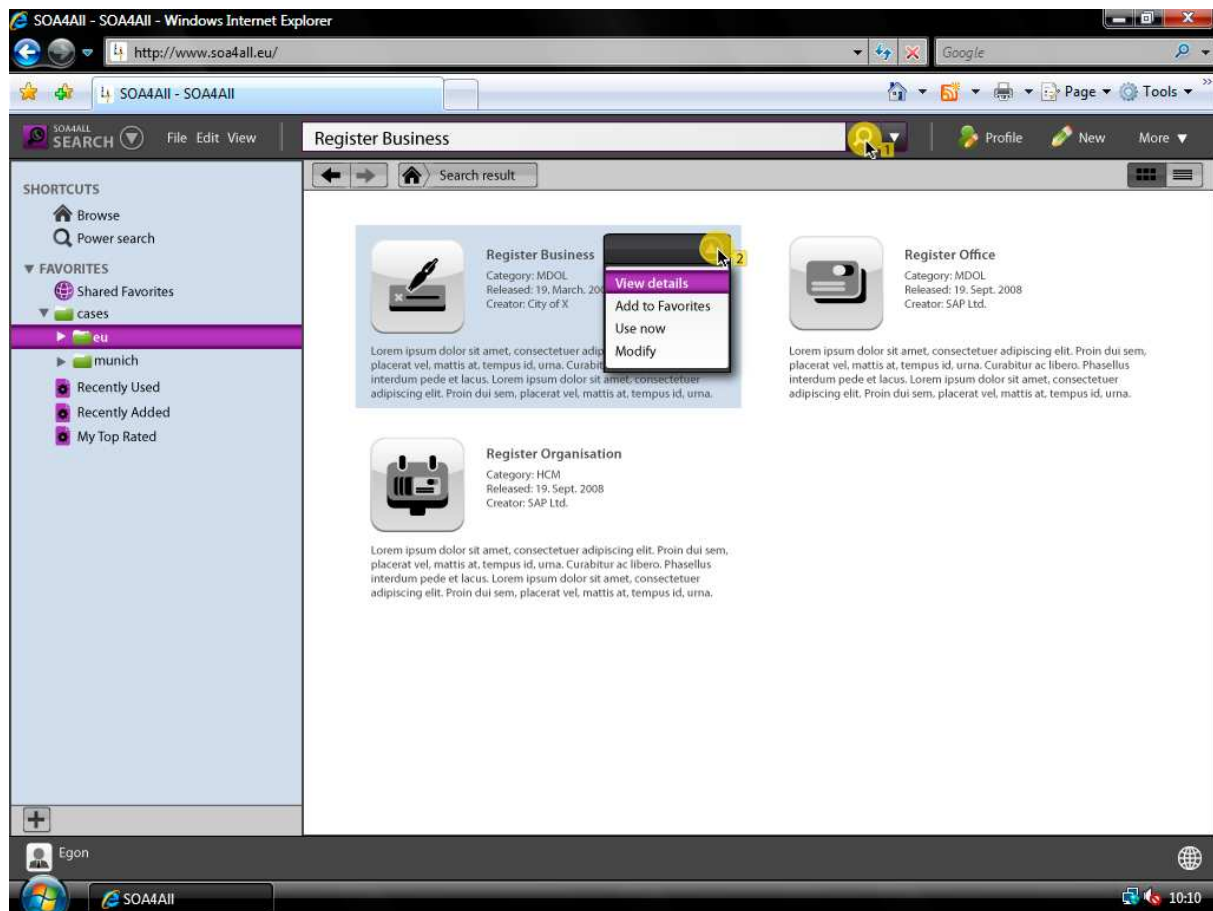
Actor: Egon

SOA4All Components

WP1 T1.3 Storage Infrastructure

WP2 T2.4 Studio, Storage Services, Search

GUI Mockup



Description

To make sure he has found the correct process, Egon switches to a details view that also shows the process annotations.

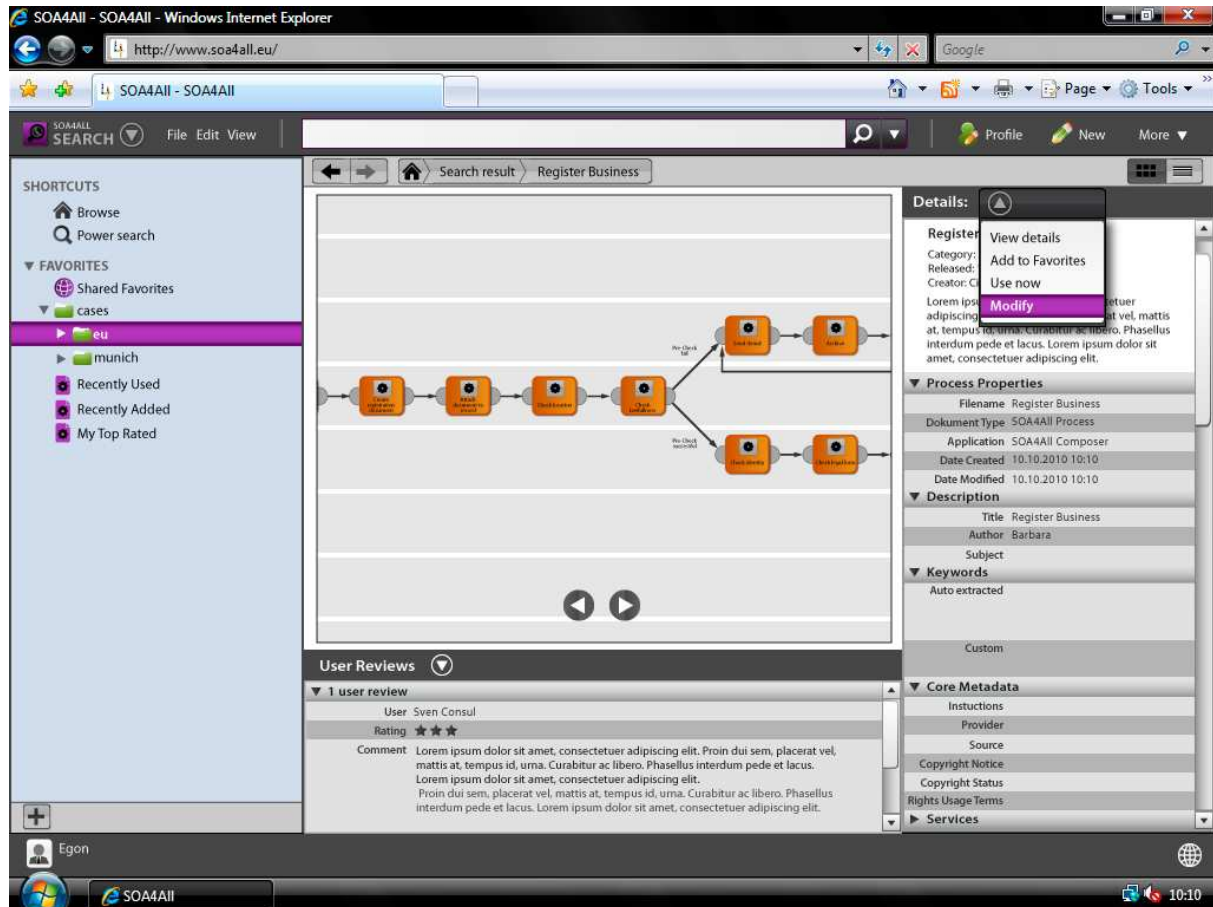
Step 6c

Actor: Egon

SOA4All Components

WP2 T2.4 Studio

GUI Mockup



Description

Egon views the details and comes to the conclusion that he has found the correct service. Thus, he switches to the SOA4All Composer.

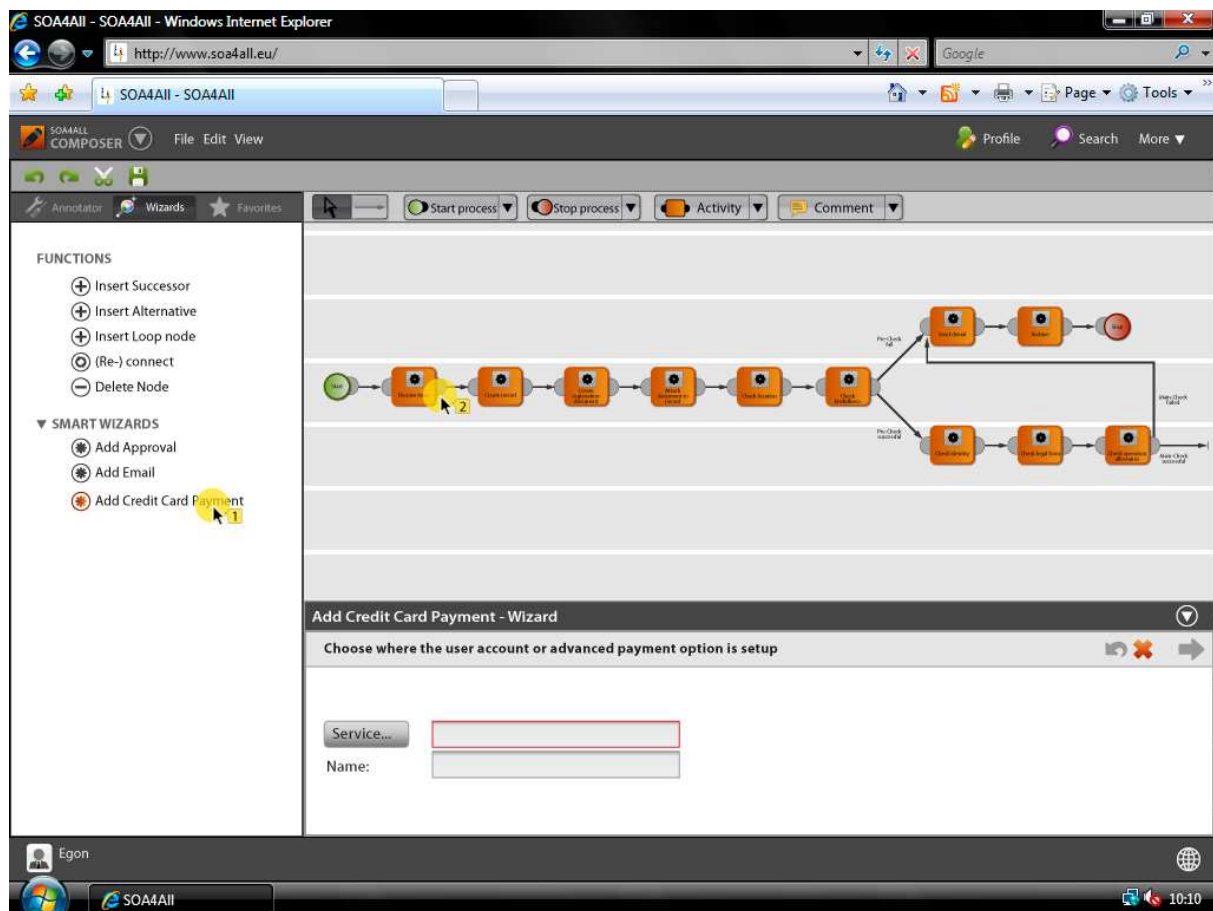
Step 7a

Actor: Egon

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor, Wizard
- T2.7 Recommender
- WP6 T6.3 Modeling Language
- WP7 T7.4 Customized Prototype

GUI Mockup



Description

Egon modifies the existing process model using a wizard that guides him through the necessary steps and does the required modifications. After selecting the appropriate wizard from the list that is automatically created based on his profile, the first step for Egon is to select the part of the process, where the payment information is entered by the user.

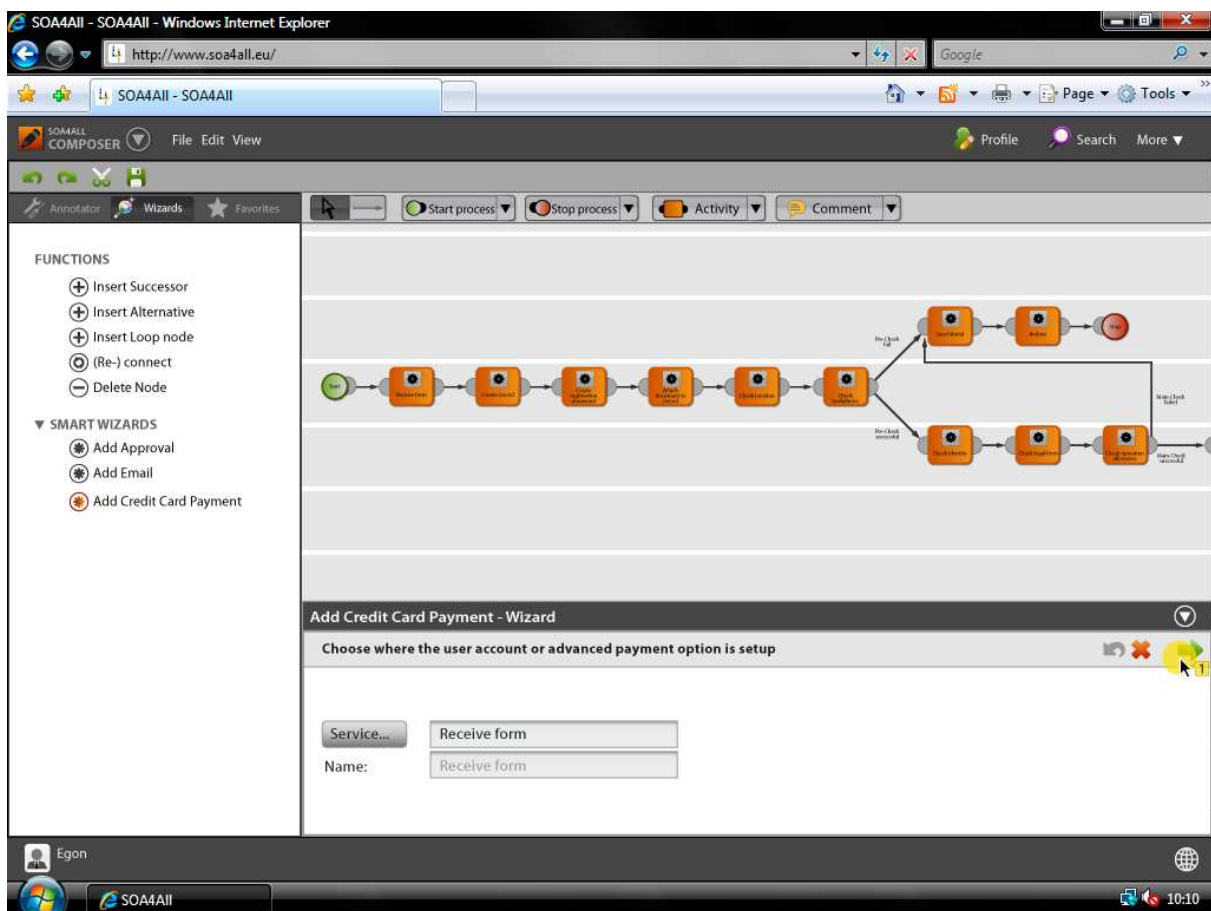
Step 7b

Actor: Egon

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor, Wizard
- WP6 T6.3 Modeling Language
- WP7 T7.4 Customized Prototype

GUI Mockup



Description

The wizard automatically fills in the selected service into the appropriate text fields.

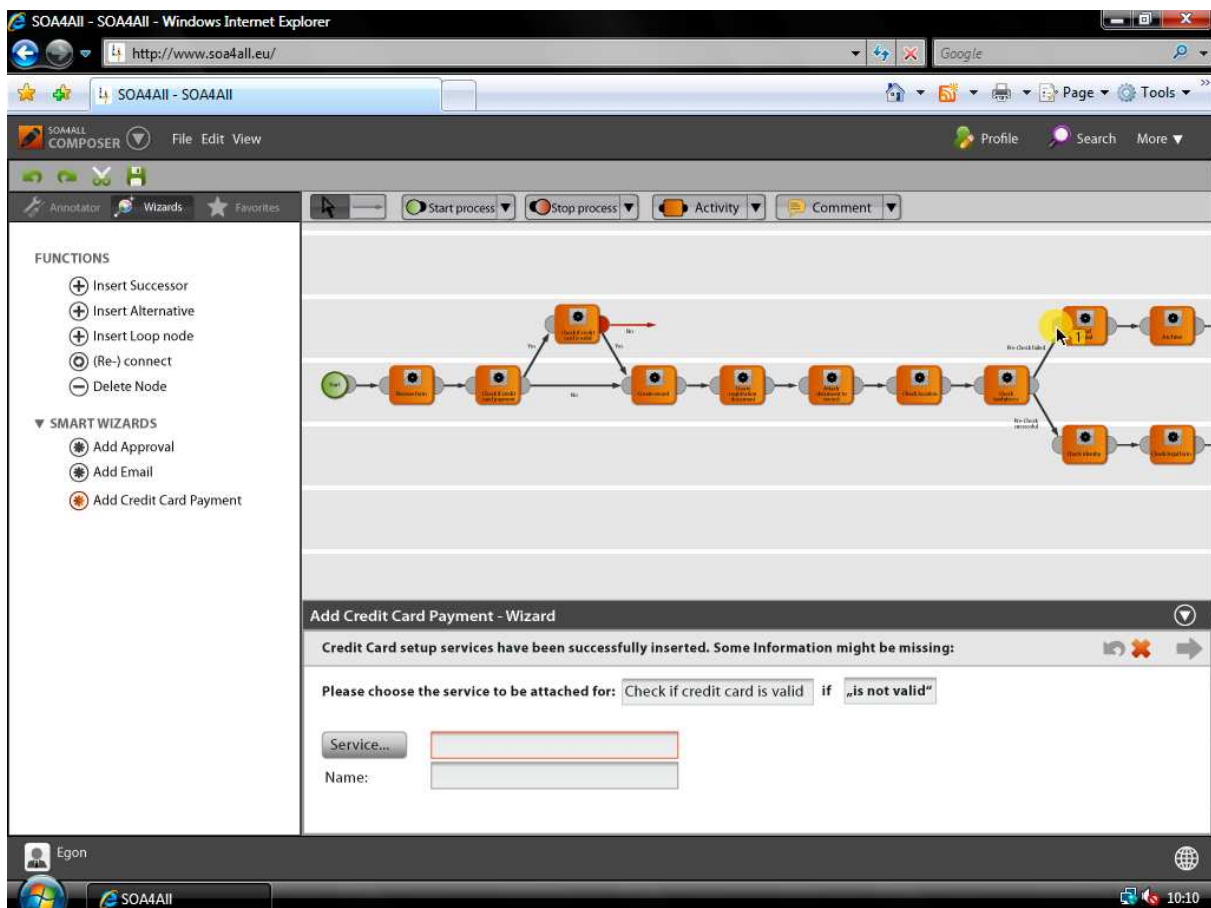
Step 7c

Actor: Egon

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor, Wizard
- WP6 T6.3 Modeling Language
- WP7 T7.4 Customized Prototype,
T7.6 Semantically Annotated Web Service

GUI Mockup



Description

Based on this information, the wizard inserts a new activity “check if credit card is valid” that is executed by calling an appropriate Web service. Because the check can either have a positive or negative result, the wizard asks Egon to define the follow up activities for each possibility.

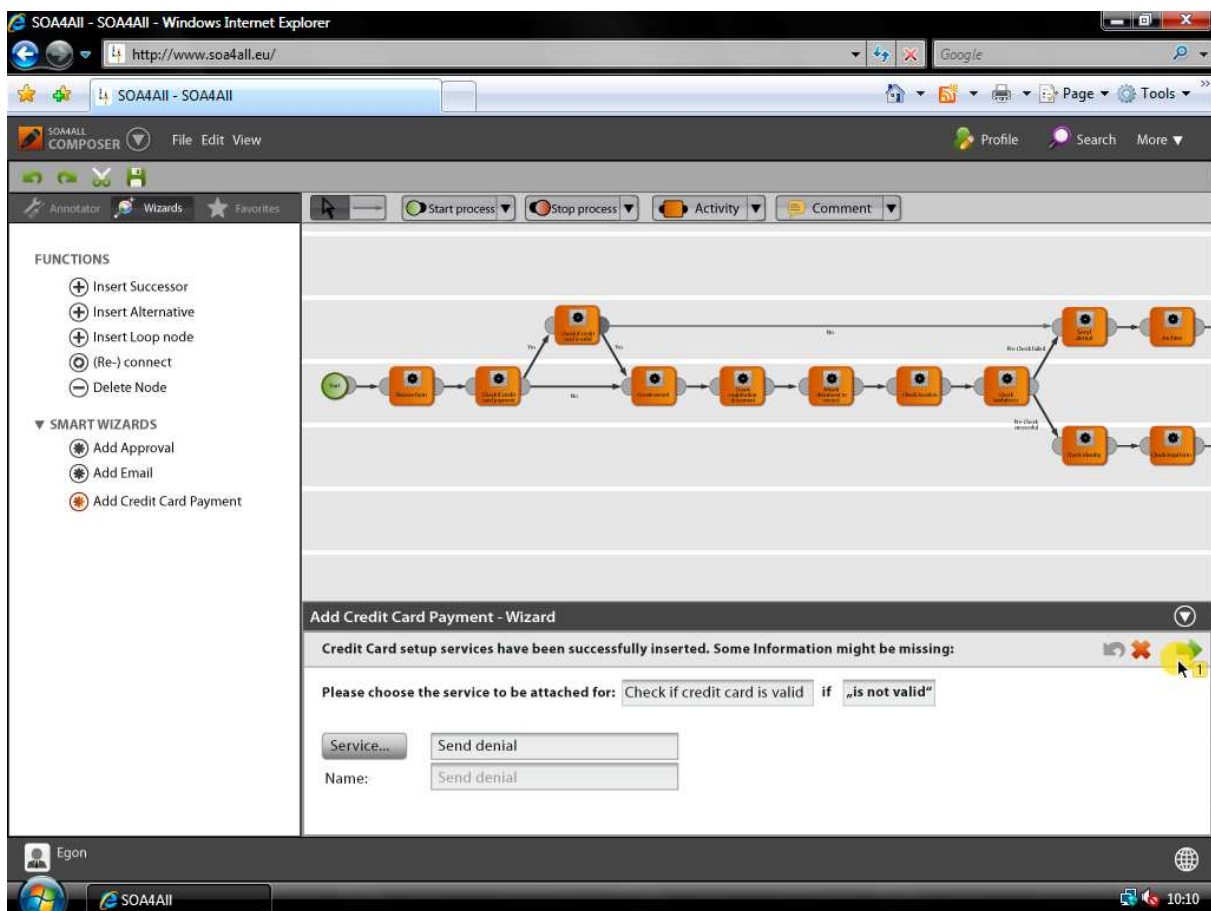
Step 7d

Actor: Egon

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor, Wizard
- WP6 T6.3 Modeling Language
- WP7 T7.4 Customized Prototype

GUI Mockup



Description

Egon does that by clicking on the desired follow-up activities. The wizard fills out the text fields automatically.

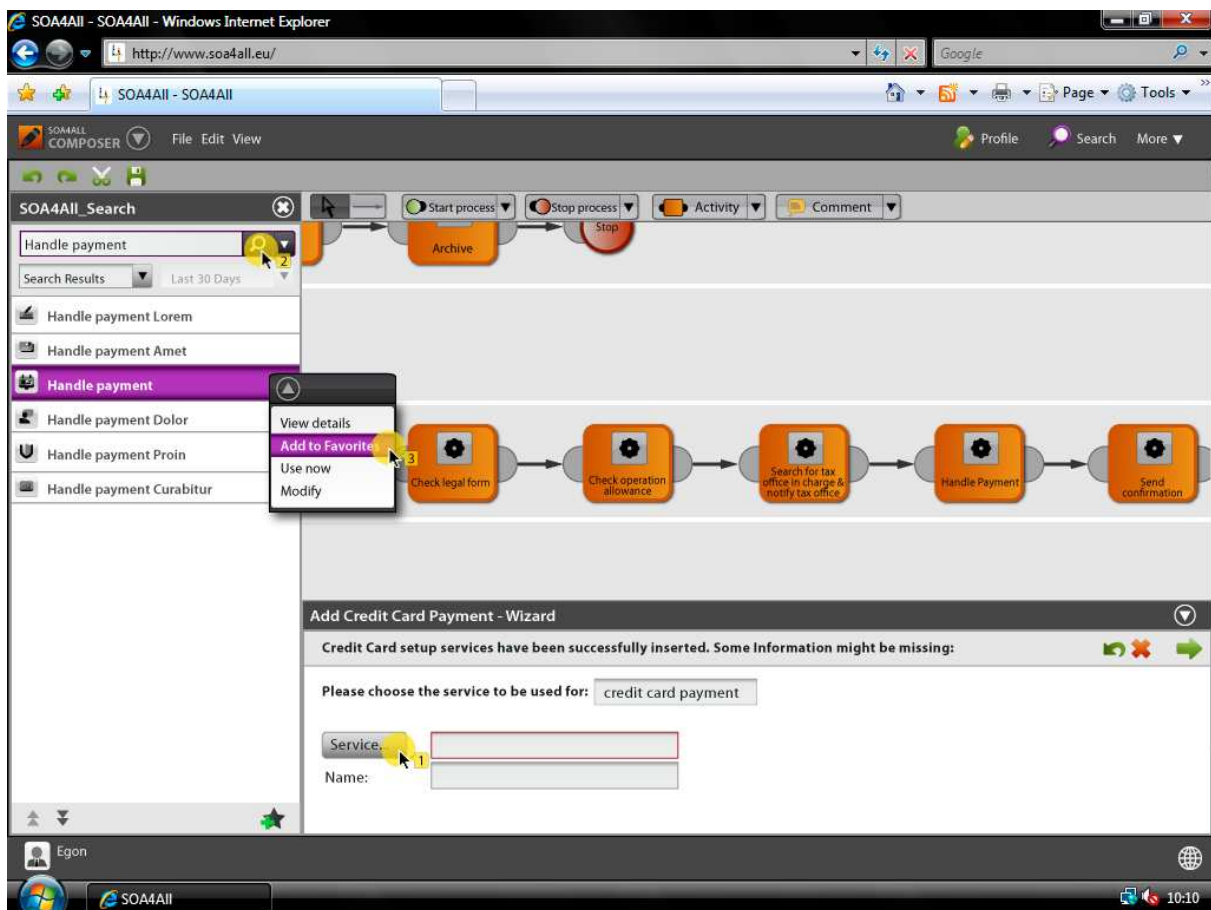
Step 7e

Actor: Egon

SOA4All Components

- WP2 T2.4 Studio
- T2.6 Process Editor, Wizard
- WP6 T6.3 Modeling Language
- T6.4 Design Time Composition
- WP7 T7.4 Customized Prototype
- T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

After the additional credit card check has been added to the process, the next step is to replace the old payment activity with a generalized one that automatically chooses between invoice and credit card payment, depending on the user's preferences: Instead of having to model the two alternative payment methods explicitly (which would also require to model the business logic to decide which payment method should be used when), Egon uses the goal "Handle Payment" that will be linked at run time to the concrete payment service. Thus, the process itself stays simple.

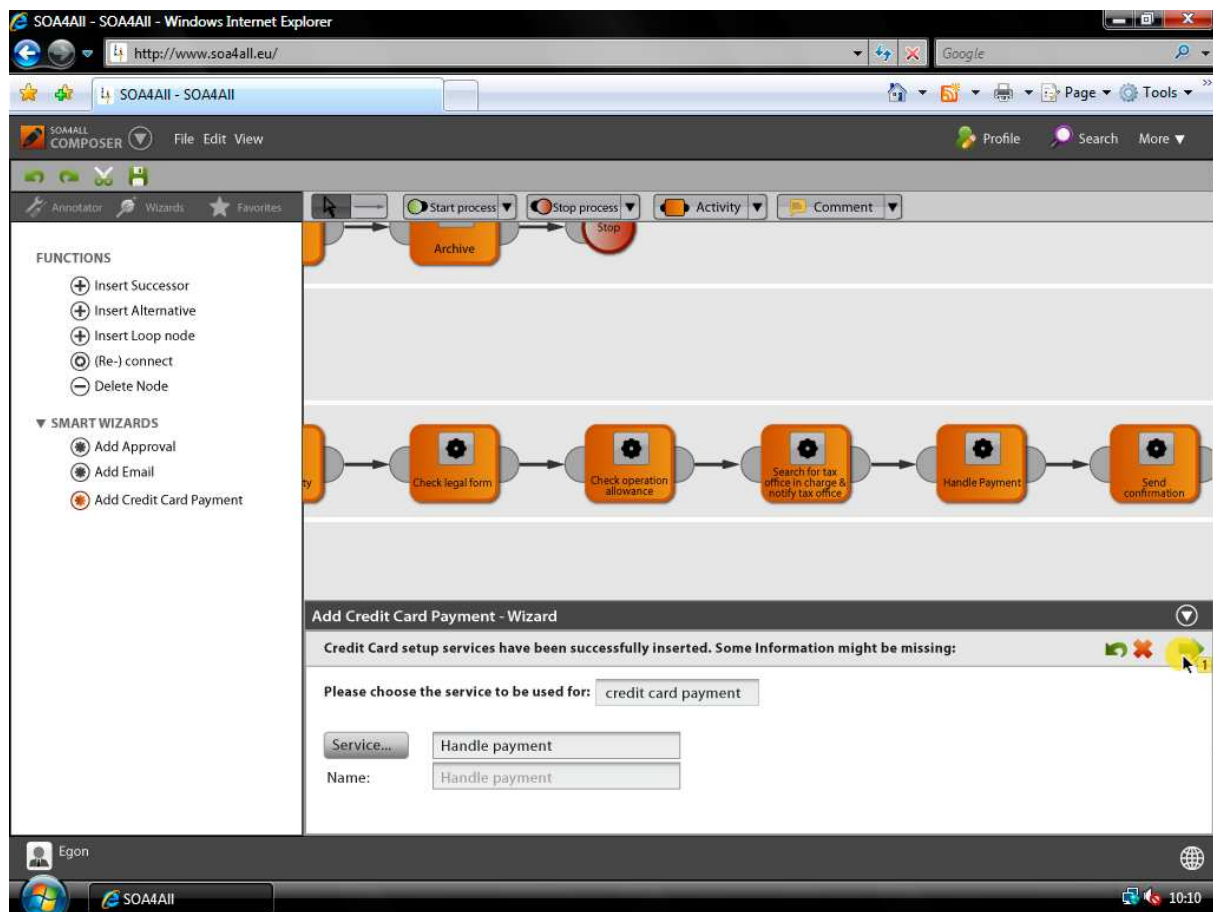
Step 7f

Actor: Egon

SOA4All Components

- WP2 T2.4 Studio
 - T2.6 Process Editor, Wizard
- WP6 T6.3 Modeling Language
 - T6.4 Design Time Composition
- WP7 T7.4 Customized Prototype
 - T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

The wizard fills in the required information automatically for Egon to review.

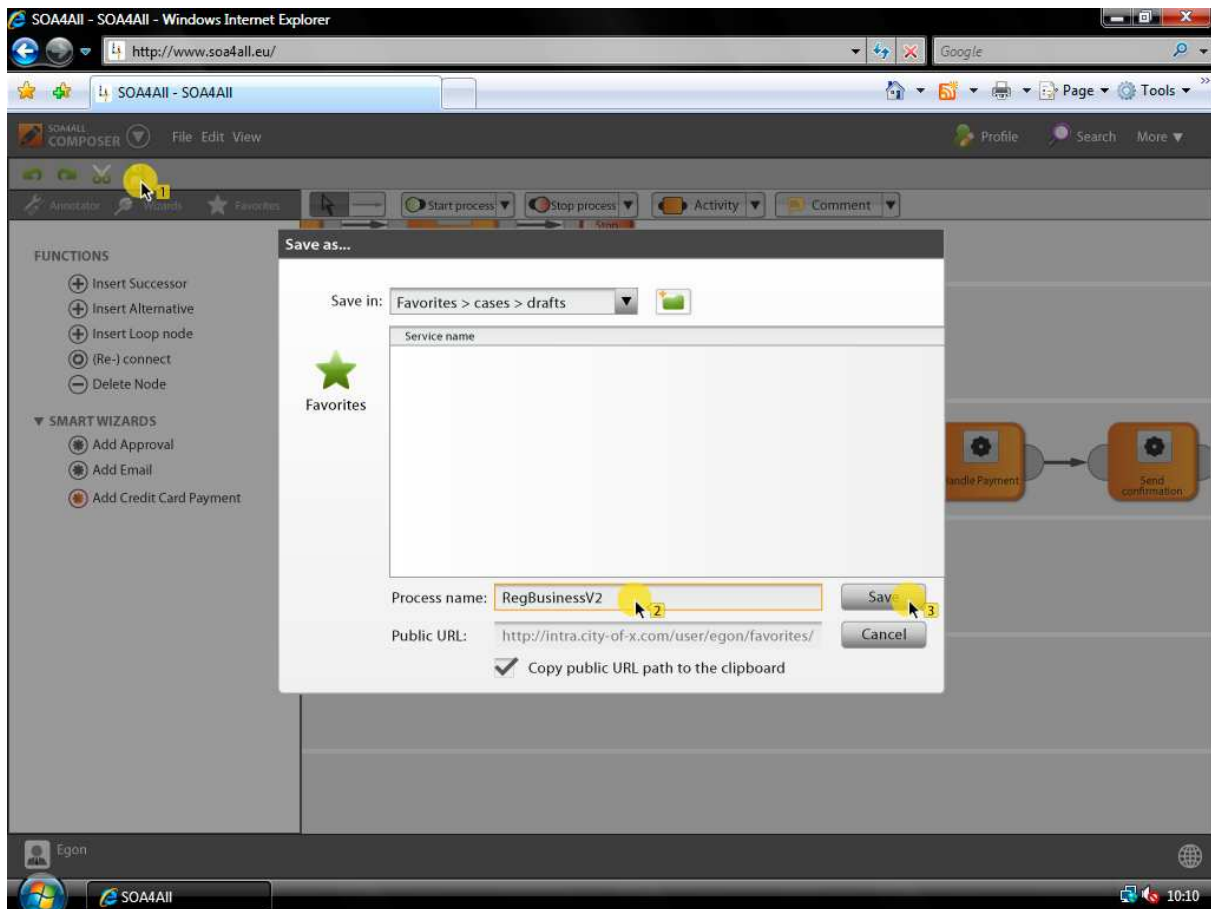
Step 8

Actor: Egon

SOA4All Components

- WP1 T1.3 Storage Infrastructure
- WP2 T2.4 Studio, Storage Services
- T2.6 Process Editor

GUI Mockup



Description

Egon saves the modified process model. Instead of replacing the original version of the process model (V1), a new version is created with a unique URI. In this example, Egon's modifications do not require an additional compliance check because they do not touch legal requirements.

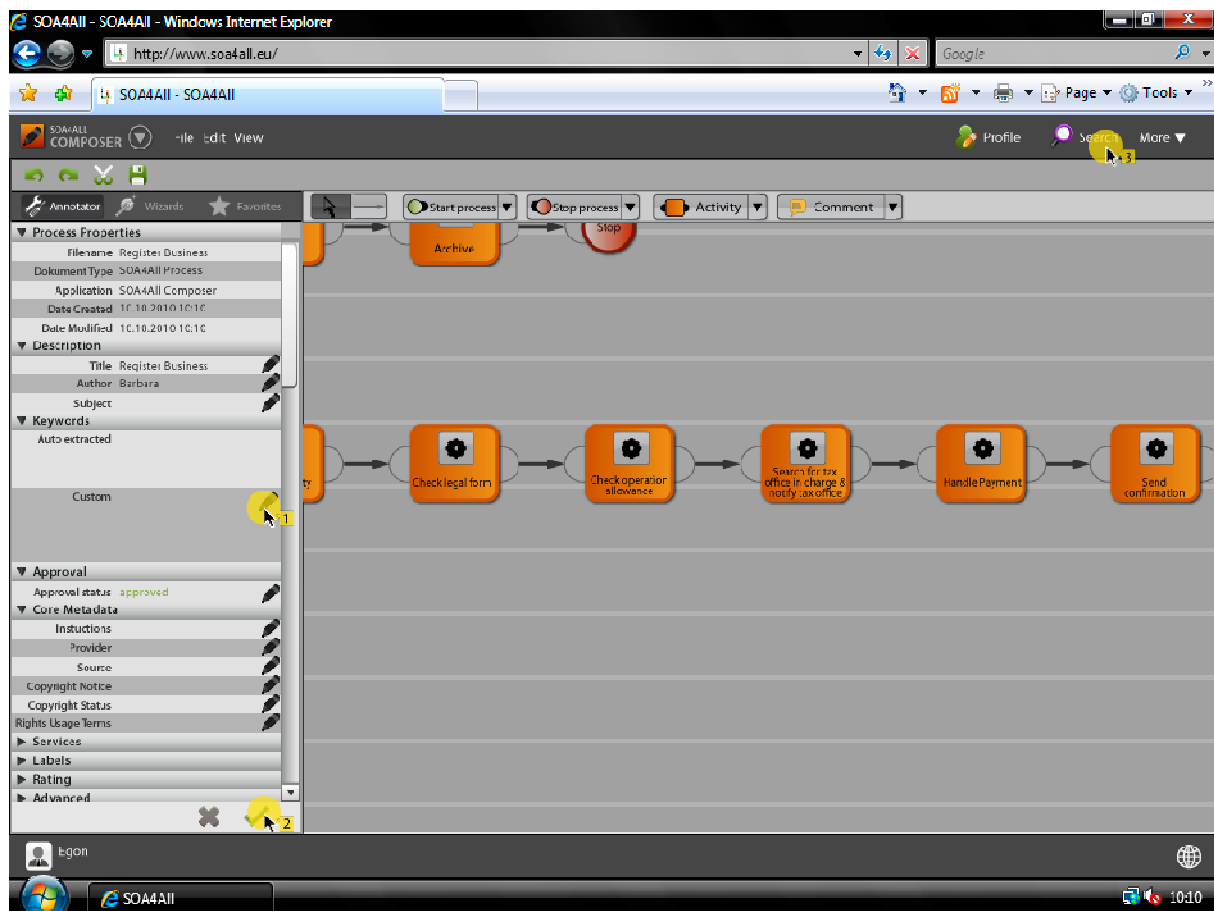
Step 9

Actor: Egon

SOA4All Components

- WP2 T2.1 Service Provisioning, Service Annotations
- T2.4 Studio
- T2.6 Process Editor
- WP3 T3.4 Semantic Service Description
- WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

Finally, Egon annotates the process model to describe his changes.

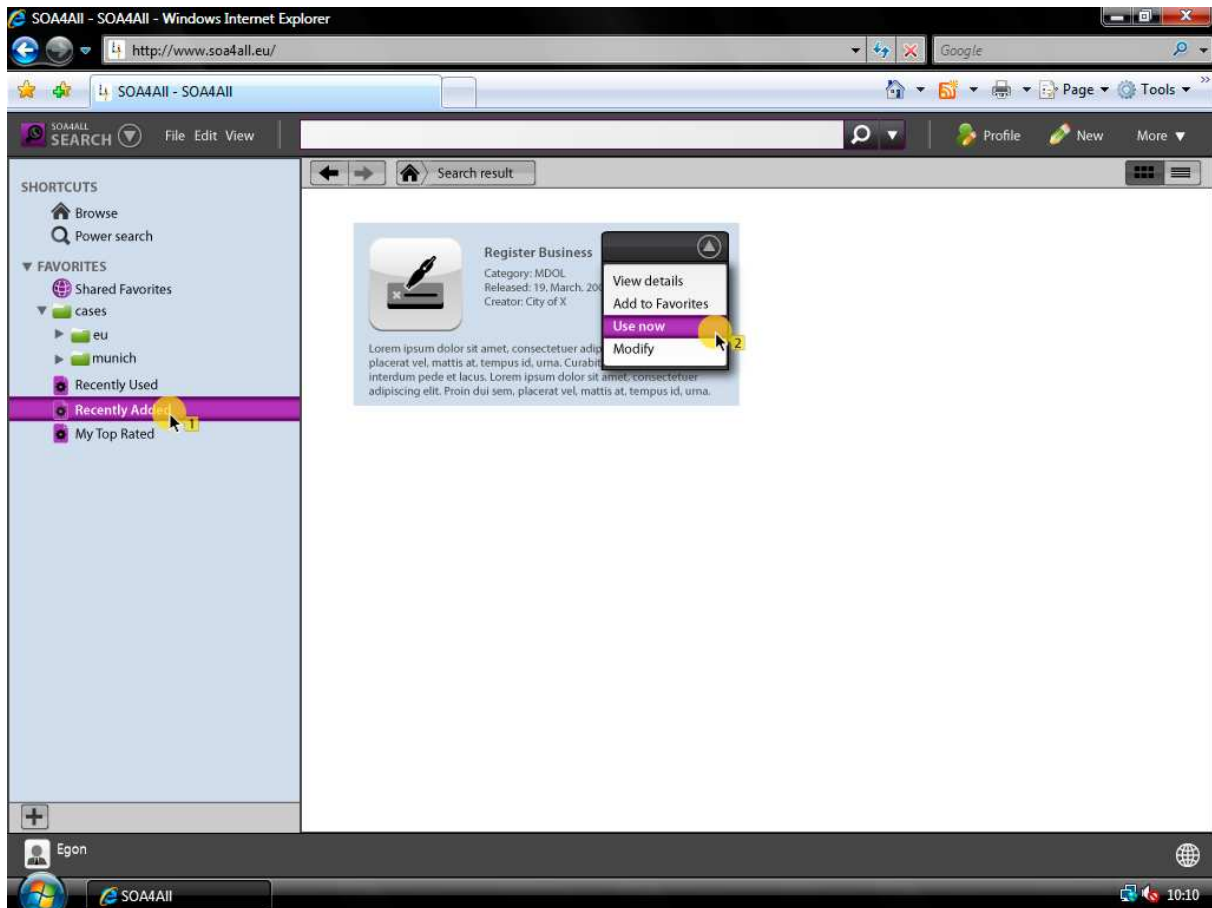
Step 10

Actor: Egon

SOA4All Components

- WP1 T1.4 Distributed Service Bus
- WP2 T2.2 Service Consumption
 - T2.4 Studio
- WP6 T6.5 Process Execution Engine
- WP7 T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

Egon executes the process so that it can handle incoming requests. Implicitly, the process is deployed to the process execution engine which has been installed within the Intranet of the City of X and which has been pre-configured as default execution engine for the service delivery platform (in other scenarios there might be several engines with an automatic load balancing mechanism in place or the user might select the engine himself).

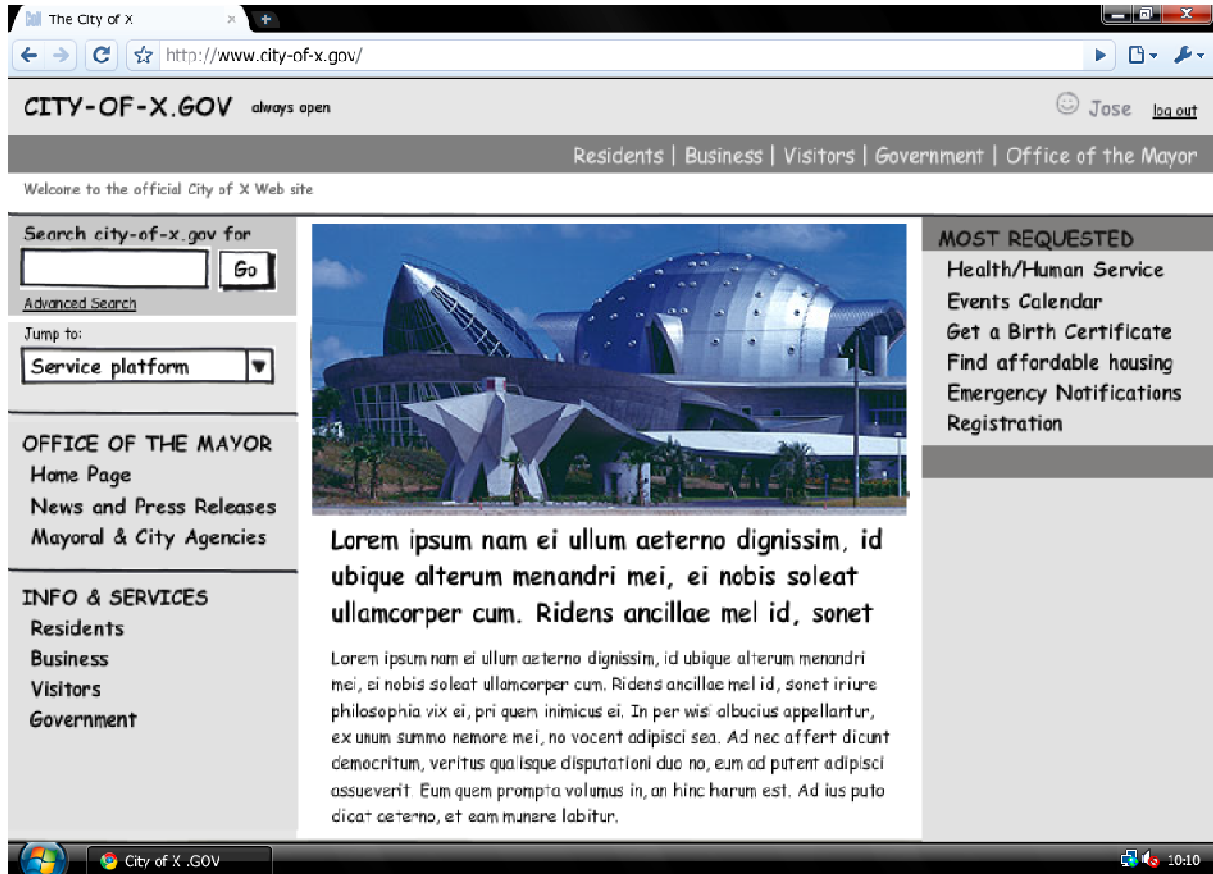
Step 11a

Actor: Jose

SOA4All Components

WP7 T7.4 Mockup of City Portal

GUI Mockup



Description

Jose accesses the Internet portal of the City of X to register his business.

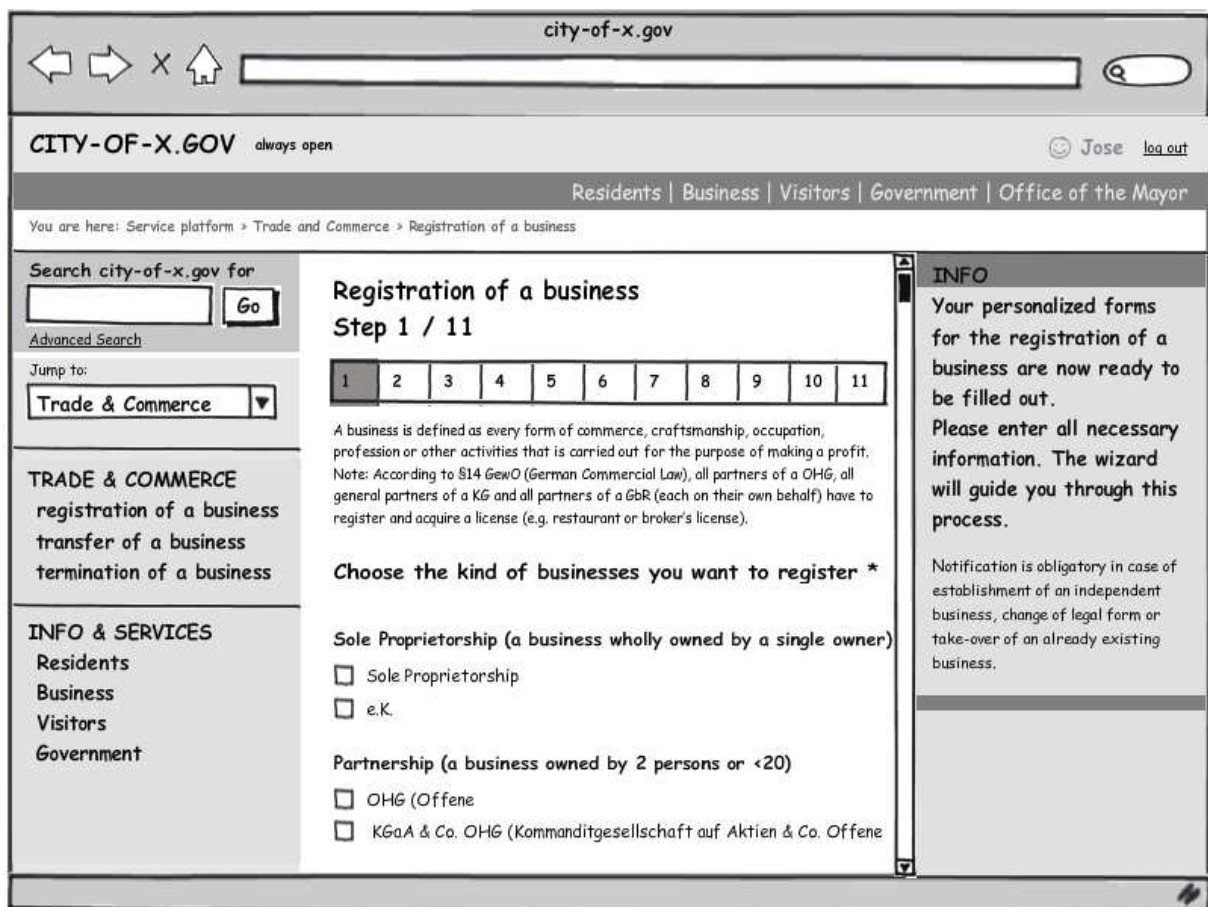
Step 11b

Actor: Jose

SOA4All Components

- WP1 T1.4 Distributed Service Bus
- WP6 T6.5 Process Execution Engine
- WP7 T7.4 Mockup of City Portal
- T7.6 Semantically Annotated SAP ES

GUI Mockup



The screenshot shows a web browser window for 'city-of-x.gov'. The page title is 'CITY-OF-X.GOV always open'. The user is logged in as 'Jose' with a 'log out' link. The navigation menu includes 'Residents | Business | Visitors | Government | Office of the Mayor'. The breadcrumb trail is 'You are here: Service platform > Trade and Commerce > Registration of a business'.

The main content area is titled 'Registration of a business Step 1 / 11'. It features a progress bar with 11 steps, where step 1 is active. Below the progress bar, there is a definition of a business and a note about German Commercial Law. The main heading is 'Choose the kind of businesses you want to register *'. There are two sections: 'Sole Proprietorship (a business wholly owned by a single owner)' with options for 'Sole Proprietorship' and 'e.K.', and 'Partnership (a business owned by 2 persons or <20)' with options for 'OHG (Offene)' and 'KGaA & Co. OHG (Kommanditgesellschaft auf Aktien & Co. Offene)'. A search bar is visible on the left, and an 'INFO' sidebar on the right provides instructions: 'Your personalized forms for the registration of a business are now ready to be filled out. Please enter all necessary information. The wizard will guide you through this process. Notification is obligatory in case of establishment of an independent business, change of legal form or take-over of an already existing business.'

Description

Jose opens the appropriate service request and enters the required information. When he sends the request, a new instance of the “Registration of a Business” process is instantiated and Jose’s data is handed over. Jose receives an electronic ticket with a unique URL so that he can access the status of his request any time.

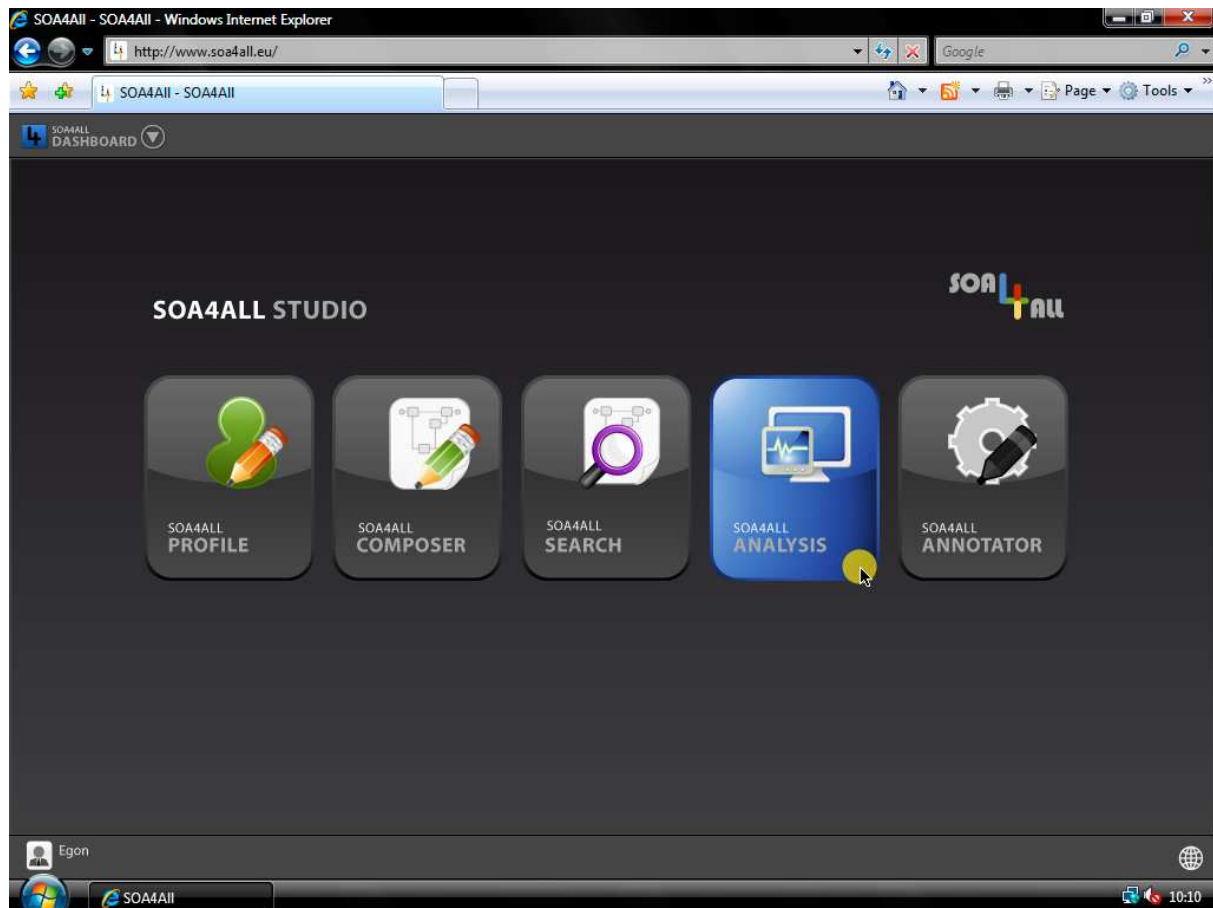
Step 12

Actor: Egon

SOA4All Components

- WP1 T1.4 Distributed Service Bus
 - T1.6 Service Monitoring
- WP2 T2.3 Service Analysis
 - T2.4 SOA4All Studio, Dashboard
- WP6 T6.5 Process Execution Engine
- WP7 T7.4 Customized Prototype
 - T7.6 Semantically Annotated SAP ES

GUI Mockup



Description

Within the services platform, Egon handles all incoming requests by constituents that are assigned automatically to him (considering his area of expertise and availability). In our example, Egon handles the process for Jose and executes the human tasks in the process (i.e., the checks defined in Figure 17, Deliverable D7.2). Logging into the service platform, Egon selects the SOA4All Analysis tool to review all active processes assigned to him.

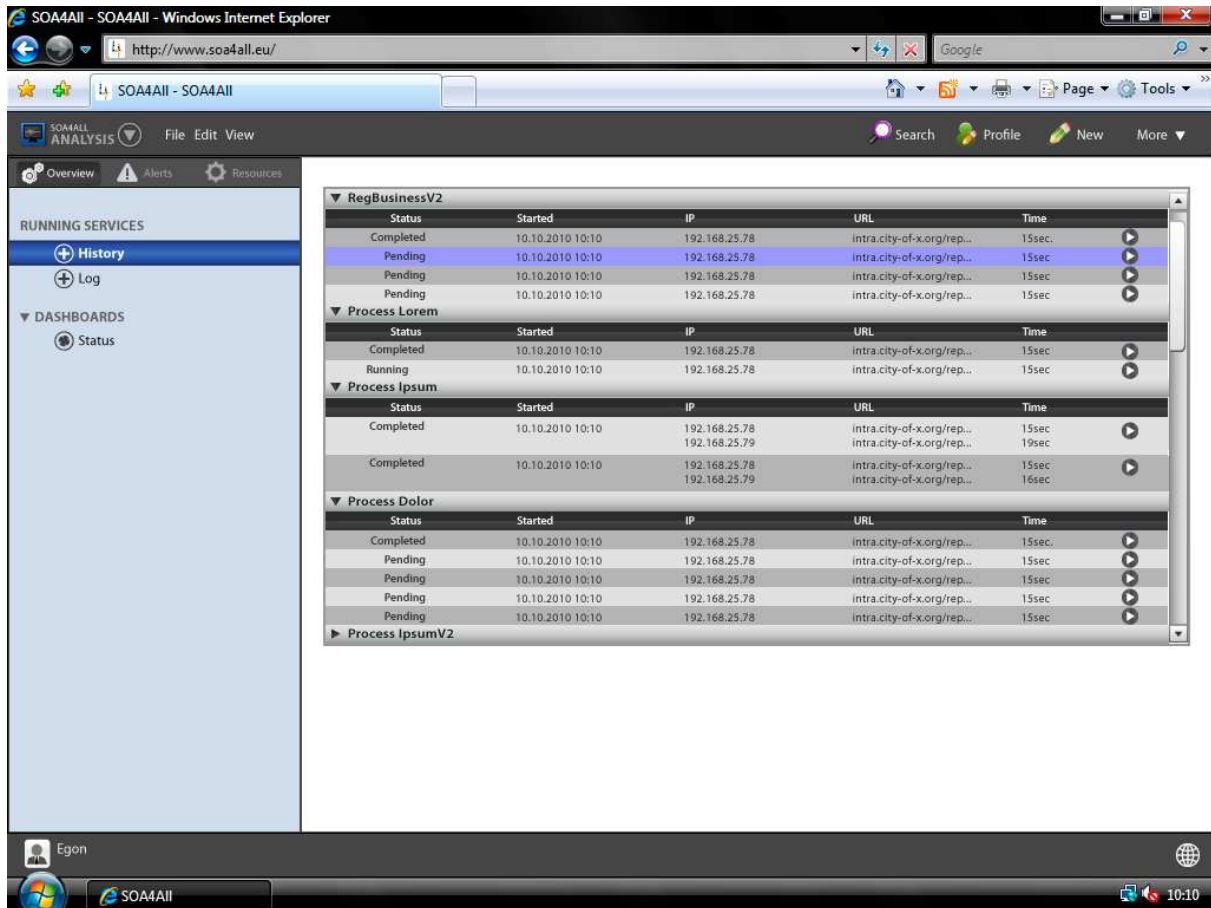
Step 13

Actor: Egon

SOA4All Components

- WP1 T1.3 Storage Infrastructure
 - T1.4 Distributed Service Bus
 - T1.6 Service Monitoring
- WP2 T2.3 Service Analysis
 - T2.4 SOA4All Studio
- WP6 T6.5 Process Execution Engine
- WP7 T7.4 Customized Prototype
 - T7.6 Semantically Annotated SAP ES

GUI Mockup



The screenshot shows the SOA4All GUI in a Windows Internet Explorer browser window. The address bar shows the URL <http://www.soa4all.eu/>. The browser title is "SOA4All - SOA4All". The application interface includes a menu bar with "File", "Edit", and "View", and a search bar. The main content area displays a list of running services and processes, organized into sections: "RegBusinessV2", "Process Lorem", "Process Ipsum", "Process Dolor", and "Process IpsumV2". Each section contains a table with columns for Status, Started, IP, URL, and Time. The "RegBusinessV2" section shows two rows: one "Completed" and one "Pending". The "Process Lorem" section shows one "Running" row. The "Process Ipsum" section shows two "Completed" rows. The "Process Dolor" section shows one "Completed" and three "Pending" rows. The "Process IpsumV2" section shows one "Pending" row. The left sidebar contains "RUNNING SERVICES" with "History" and "Log" buttons, and "DASHBOARDS" with a "Status" button. The bottom status bar shows the user "Egon" and the time "10:10".

Description

The Analysis tool gives Egon an overview of all running processes. If desired, Egon can select one to check its status or to execute a human task.

Step 14

Actor: -

SOA4All Components

- WP1 T1.4 Distributed Service Bus
- WP6 T6.5 Process Execution Engine
- WP7 T7.4 Customized Prototype
- T7.6 Semantically Annotated SAP ES

Description

Once the manual checks are successfully completed, the rest of the process is executed automatically (see Figure 17, Deliverable D7.2). The payment method is selected depending on the information (context) provided by Jose at Step 11b. Finally, the confirmation (approval) is send to Jose and the process is complete.

Annex B: Examples for Service Annotations

```

<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:creditcheck="http://www.deltavista.at/schema/cWreditcheck"
  xmlns:sawsdl="http://www.w3.org/ns/sawsdl#" targetNamespace="http://www.deltavista.at/schema/creditcheck">
  <types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.deltavista.at/schema/creditcheck"
      elementFormDefault="qualified">
      <element name="individualCreditCheckRequest" type="creditcheck:IndividualCreditCheckRequest"
        sawsdl:modelReference="http://example.com/hcc#DeltavistaUser"
        http://example.com/onto#CurrentTime
        http://example.com/hcc#Individual"
        sawsdl:loweringSchemaMapping="http://example.com/hanival-individual-check-lowering.xsparql" />
      <element name="individualCreditCheckReply" type="creditcheck:IndividualCreditCheckReply"
        sawsdl:liftingSchemaMapping="http://example.com/hanival-xml-lifting.xsparql" />
      <element name="companyCreditCheckRequest" type="creditcheck:CompanyCreditCheckRequest"
        sawsdl:modelReference="http://example.com/hcc#DeltavistaUser"
        http://example.com/onto#CurrentTime
        http://example.com/hcc#Company"
        sawsdl:loweringSchemaMapping="http://example.com/hanival-company-check-lowering.xsparql" />
      <element name="companyCreditCheckReply" type="creditcheck:CompanyCreditCheckReply"
        sawsdl:liftingSchemaMapping="http://example.com/hanival-xml-lifting.xsparql" />
      <complexType name="ServiceRequest" abstract="true">
        <sequence>
          <element name="credentials" type="creditcheck:Credentials" />
          <element name="user" type="creditcheck:User" minOccurs="0">
            <annotation>
              <documentation>This optional element can be used to identify
                and describe the requesting user. If provided, this
                information is displayed in the web ui.</documentation>
            </annotation>
          </element>
          <element name="reference" type="string" minOccurs="0">
            <annotation>
              <documentation>This optional field can be used to associate a
                reference with the request. This reference is searchable
                in the archive. It need not be unique.</documentation>
            </annotation>
          </element>
          <element name="correlationId" type="string" minOccurs="0">
            <annotation>
              <documentation>This field is simply passed through to the
                reply. It can be used to correlate request and
                response.</documentation>
            </annotation>
          </element>
        </sequence>
      </complexType>
    </schema>
  </types>

```

Listing 1: SAWSDL Annotation Applied to a WSDL Definition

```
namespace {_"http://www.example.org/ontologies/example#",
  dc _"http://purl.org/dc/elements/1.1#",
  foaf _"http://xmlns.com/foaf/0.1/",
  wsmo _"http://www.wsmo.org/wsmo/wsmo-syntax#",
  loc _"http://www.wsmo.org/ontologies/locationOntology#",
  general _"http://example.com/onto",
  ccheck _"http://example.com/hcc" }

goal _"http://example.org/CreditCardCheckGoal"
  importsOntology {_"http://www.wsmo.org/ontologies/locationOntology", _"http://example.com/hcc" }
  capability
    postcondition
      definedBy
        ?cardUser[
          userName hasValue ?UserName,
          userGender hasValue ?UserGender,
          userAge hasValue ?UserAge
        ] memberOf ccheck#DeltavistaUser
      and
      (
        ?cardInformation[
          type hasValue "PremiumCard"
        ] memberOf ccheck#CreditCardInformation
      or
        ?cardInformation[
          type hasValue "GoldCard"
        ] memberOf ccheck#CreditCardInformation
      )
      and ?cardValidityStatus memberOf ccheck#validityStatus
      and ?bank memberOf ccheck#PrivateCustomerBank
      and
        ?currentTime[
          timeValue hasValue ?dateTimeCheck
        ] memberOf general#DateTime.
```

Listing 2: WSMO Goal for Checking Credit Card Validity