D3.2 Functional Specification

Authors: INESC (Lead), UVI, TUDA, UVA, ASC, TIE, ISOFT

Delivery Date:	Due Date:	Dissemination Level:
2012-06-06	2012-04-30	Public

This document contains the functional specification of the ADVENTURE platform. It defines the functionalities of the platform and shows, from a user perspective, how its components fit together in order to provide and intuitive and user-friendly interface for the management of the virtual factories. It also shows, from a process perspective, how the components of the platform interact in order to execute, orchestrate, monitor, forecast and adapt the dynamic manufacturing processes of the virtual factories.

SEVENTH FRAMEWORK PROGRAMME

Document History	
Draft Version	V0.1, INESC, November 16th, 2011 V0.2, INESC, March 16th, 2012 V0.3, INESC, March 18th, 2012 V0.4, INESC, March 20th, 2012 V0.5, INESC, March 29th, 2012 V0.6, INESC, April 1st, 2012 V0.7, INESC, May 2nd, 2012 V0.8, INESC, TIE, UVA, TUDA, May 3rd, 2012 V0.9, INESC, TIE, ISOFT, ASC, TUDA, UVA, May 11th, 2012 V1.0, INESC, TIE, ISOFT, ASC, TUDA, UVA, UVI, May 23th, 2012 V1.1, INESC, TIE, ISOFT, ASC, TUDA, UVA, UVI, May, 24th, 2012 V1.2, INESC, May, 25th, 2012 V1.3, INESC, May, 28th, 2012 V1.4, INESC, May, 29th, 2012 V1.5, INESC, May, 29th, 2012 V1.6, INESC, May, 29th, 2012 V1.7, TIE, INESC, May, 30th, 2012 V1.8, INESC, TIE, ISOFT, ASC, TUDA, UVA, UVI, June, 1st, 2012 V1.9, TANet, INESC, June, 5th, 2012 V2.0, TUDA, June, 6th, 2012
Contributions	INESC PORTO - Américo Azevedo - Filipe Ferreira - José Faria ISOFT - Atanas Manafov - Georgi Pavlov - Irena Pavlova - Velimir Manafov TUDA - Dieter Schuller - Ronny Hans - Sebastian Zöller UVI - Juergen Mangler - Tobias Hildebrandt TIE - Juanvi Vidagany - Lennert Kujipers - Stuart Campbell ASC - Rafael Karbowski - Sven Abels - Tim Dellas UVA - Ahm Shamsuzzoha - Yuqiuqe Hao
Internal Review 1	Gash Bhullar, Simon Osborne, TANet
Internal Review 2	Stuart Campbell, TIE
Internal Review 3	Stuart Campbell, TIE, and Gash Bhullar, TANet

Final Version	June 6th, 201

Table of Contents

List of Figures

List of Tables

Executive Summary

This document contains the functional specification of the ADVENTURE platform developed in the context of task T3.2.

Sections 1 reviews the project aims, presents the scope and target audience of the document.

Section 2 provides a system overview according to what was defined in previous project deliverables.

Section 3 contains a system level functional analysis that shows how the components of the platform cooperate in order to provide the desired functionality to the end user. This is a fundamental section since it provides a common understanding of the platform operation and provides the context for the detailed specification of the components presented in section 5. The functional analysis is organized according to two perspectives: user perspective and process execution. The user perspective shows how the components of the platform fit together in order to provide an intuitive and user-friendly interface. The process execution perspective shows how the components of the platform interact in order to instantiate, orchestrate, monitor, forecast and adapt virtual factory manufacturing processes.

Section 4 presents the use case scenarios that will be used for system verification.

Section 5 provides a component level specification and contains a subsection for each component of the platform addressing the following topics:

- **Overall functional characterization.** This provides a short summary of the main functionalities implemented by the component
- **Use cases.** This identifies and describes the use cases implemented by the component as well as user interface mockups.
- **Component interaction**. This presents the internal structure of the component and describes in detail the interactions with the other components of the platform
- **Conceptual data model**. This presents a preliminary analysis of the data structure that will support the component.
- **Parameters to take into account in the technical specification**. This section contains a table with generic and component specific criteria which will be acting as an input to technical specification for selecting technologies.

Section 6 presents the conclusions and perspectives of further work.

Annex A shows the mapping between requirements identified in Task T2.3 and the functionalities specified in this deliverable.

1 Introduction

ADVENTURE – ADaptive Virtual ENTerprise manufacTURing Environment – is a project funded in the Seventh Framework Programme by the European Commission. ADVENTURE creates a framework that enhances the collaboration between suppliers, manufacturers and customers for industrial products and services. Section 1 sets the scene for the report defining aims and objectives.

1.1 ADVENTURE Project Aims

The framework proposed by ADVENTURE provides mechanisms and tools that facilitate the creation and operation of manufacturing processes in a modular way. ADVENTURE combines the power of individual factories to achieve complex manufacturing processes. It provides tools for partner-finding, process creation, process optimization, information exchange as well as real-time monitoring combined with the tracking of goods and linking them to Cloud services.

There have already been several research projects that address the combination of different independent manufacturers to so-called virtual factories. Most of these research projects focus primarily on the business-side in general and on aspects like partner-finding and factory-building processes in special. However no proven tools or technologies exist in the market that provide the creation of virtual factories applying end-to-end integrated Information and Communication Technology (ICT). ADVENTURE is aiming to provide such tools and processes that will help to facilitate information exchange between factories and move beyond the boundaries of the individual enterprises involved. The collaborative manufacturing process will be optimised by enabling the integration of factory selection, forecasting, monitoring, and collaboration during runtime.

ADVENTURE builds on concepts and methods of Service-oriented Computing and benefits from the advancements in this field. The monitoring and governance of the collaborative processes will be supported by technologies from the Internet of Things such as wireless sensors. Existing tools and services that can be integrated will be considered during the development of the platform for ADVENTURE.

The increased degree of flexibility provided through ADVENTURE will benefit SMEs especially as it helps them to react quickly to changes and to participate in larger, cross-organizational manufacturing processes. Furthermore, ADVENTURE will help manufacturers in assessing the environmental friendliness of actual manufacturing processes and resulting products and services. Other objectives of ADVENTURE include research in areas such as service-based manufacturing processes, adaptive process management, process compliance, and end-to-end-integration of ICT solutions.

1.2 Deliverable Purpose, Scope and Context

The purpose of this deliverable is to present the functional specification of the ADVENTURE platform at two levels:

- Section 3 presents the system functional analysis of the platform showing how all components cooperate in order to provide the desirable functionality to the end user
- Section 5 contains an in-depth analysis and specification of the functionality and behavior of each component

As shown in Figure 1, this deliverable is closely related with previous deliverables of the project, namely *D2.3 – User Requirements Report* and *D3.1 – Global Architecture Definition*.

The first of these documents specifies the functionality that the platform should provide from a user perspective, whereas the second one describes the internal components of the platform. In order to ensure a close alignment of the functional specification with the previous deliverables, this document has a specific section for each component identified in the architecture, which includes a mapping between the functionality of the component and the user requirements it supports.

The specification of the components will be platform independent but, even so, it will contain the necessary information for the technical design of the platform. Therefore, this deliverable serves as a basis for the technical specification task (T3.3), and for WP4, WP5, and WP6, where the software components that implement these functionalities will be developed.

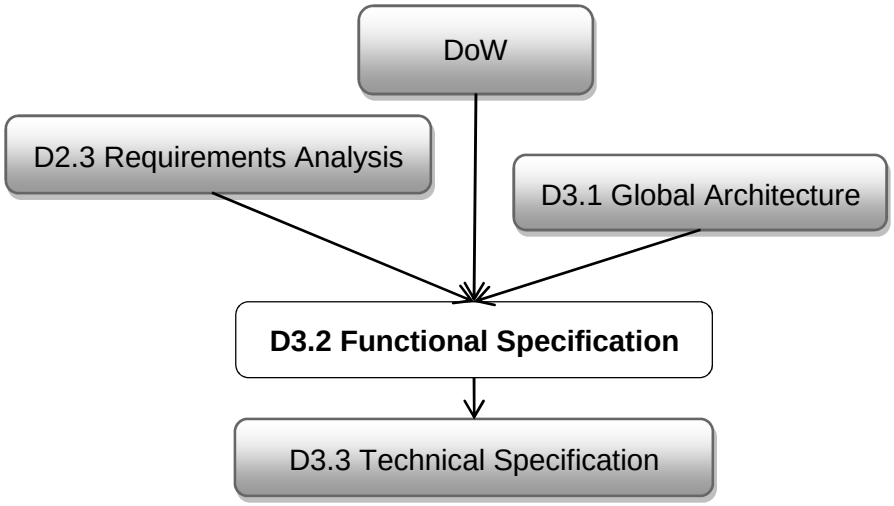

Figure 1 - Context of the Deliverable

1.3 Document Status

This document is listed in the DOW as 'public' as the functional specification may be used by external parties as a base for getting insights about the functionality of the components of ADVENTURE.

1.4 Target Audience

The document's primary audience is all participating project members, but has a special focus on those partners that are planned to be involved in technical design and software

development tasks. Particularly, software developers should always have in mind the specifications contained in this document, as the components that they will implement should meet all the features and functions described here. As a public deliverable it could also be useful for Future Internet Enterprise Systems (FINES) cluster participants and other similar projects.

1.5 Abbreviations and General Terms

A definition of common terms and roles related to the realization of ADVENTURE as well as a list of abbreviations is available in the supplementary document "Supplement: Abbreviations and General Terms" which is provided in addition to this deliverable.

Further information can be found at: http://www.fp7-ADVENTURE.eu/glossary.

1.6 Document Structure

This document is organized in 6 main sections:

Section 1 recalls the aims of the ADVENTURE project, introduces the purpose, scope, context, status and target audience of the document, and describes its organization.

Section 2 provides an updated system overview.

Section 3 presents a system level analysis of the platform from a functional oriented perspective. It shows how the components of the platform cooperate in order to provide the desired functionality to the end user at the dashboard.

Section 4 describes the scenarios that will be employed for verification of the completeness of the functional specification and its alignment with the user requirements. Later, these same use case scenarios will be considered in the validation of the software components developed in WP4, WP5 and WP6.

Section 5 presents the detailed functional specification of the platform and it contains a subsection for each component. These subsections are organized according to the following set of topics:

- Overall functional characterization, where a comprehensive description of the component is presented;
- Use cases , where the external functionality is detailed;
- Component interaction, where the internal structure and the interactions with the other components of the platform are described in detail;
- Conceptual data model, where a preliminary analysis of the data structure that will support the component is presented;
- Parameters to take into account in the technical specification, containing a table with generic and component specific criteria which will be acting as an input to technical specification for selecting technologies.

Section 6 summarizes the content of the deliverable.

Annex A contains for each component of the system two tables showing the mapping between the functionality offered by the component and the requirements identified in D2.3 (requirements Analysis Report) that apply to the component.

2 System Overview

As specified in deliverable *D3.1 Global Architecture Definition Document,* the functionalities of ADVENTURE are provided by 13 components organised in four architectural layers as shown in Figure 2.

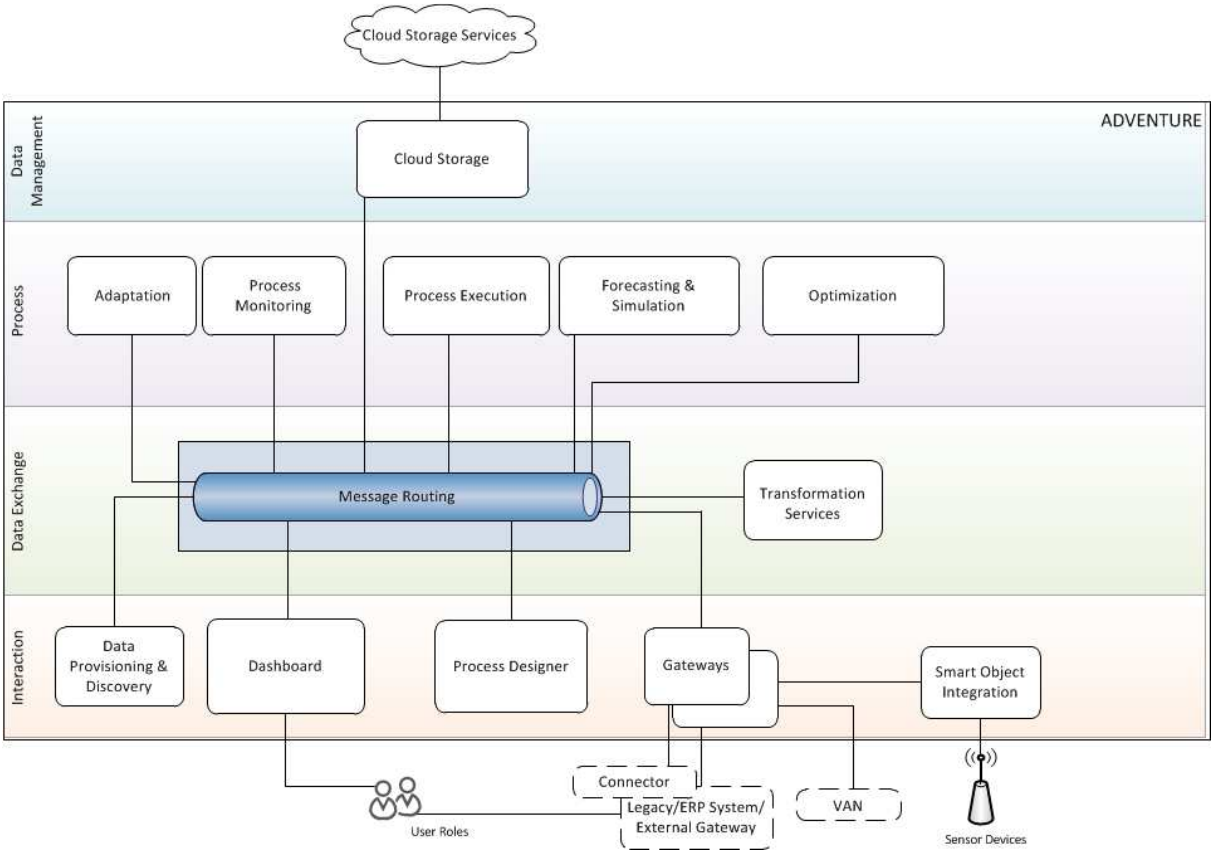

Figure 2 - Architecture of the ADVENTURE Platform

2.1 Data Management Layer

This layer contains the Cloud Data Storage service component that realizes the storage of different data types. The Cloud-based Data Storage defines and implements a data repository, where information from different sources, as e.g., from Smart Objects, will be integrated and pre-processed. The access to this repository is realised through a role model.

2.2 Process Layer

This layer hosts all the components that interact with manufacturing smart processes. The Process Execution component will be at the heart of ADVENTURE platform, as it will orchestrate all interaction in a virtual factory. The Process Execution component implements and provides functionalities which are required for executing the modelled Smart Process. It further allows to plug-in other functionalities as, e.g., forecasting and simulation, through a graphical user interface.

Forecasting & Simulation of the process comprises the execution of the process in forecast mode, monitoring the forecast / simulated results and displaying them to the user. For this, it takes information from different sources into account and integrates them. It allows a step-by-step execution of the manufacturing process. After the execution of each manufacturing step, the Adaptive Process Execution component will decide about the execution of the next step based on (recognised) unexpected events such as delays in the manufacturing processes, strikes, delays in the transportation, etc. and on monitored results from the monitoring component.

Within the Optimization component an optimization model for manufacturing processes will be conceptualized and formalized. Depending on specific optimization needs for certain processes, appropriate optimization techniques will be implemented and applied. Once the process definition has been finished and an optimized assignment of manufacturing services to the different process steps of the Smart Process has taken place, the execution of this process can be started.

The Process Monitoring component provides monitoring and functionalities for enabling up-to-date, real-time diagnosis of the process execution status required for enabling potentially necessary adaptations to the Smart Process. Additionally it can query Smart Object data sources like sensors etc. for their current state.

2.3 Data Exchange Layer

This layer provides platform wide communication services. To ensure a simple and standardized communication mechanism, a Message Routing component is needed which connects the different components and which manages the resultant communications between them. This component routes valuable information about current status of production and delivery, e.g., the incoming and dispatch of products from a certain manufacturing area. The data imported through the Gateways can be transformed into a different data format by means of the Transformation Services**.** The Transformation Services will provide the ADVENTURE users with advanced data transformation features, helping them to collaborate regardless of the internal data types they use.

2.4 Interaction Layer

This layer contains the components that interact with users and external systems, like legacy systems, VANs (value added networks), ERP systems and smart objects. To establish processes between companies, data about the partners wishing to collaborate in a virtual factory is needed. Therefore, each ADVENTURE member needs to be able to add data about their company, products, services, and processes. To achieve this in a user-friendly way, ADVENTURE the Data Provisioning & Discovery component will provide an editor to enter, view, update, or delete this data.

Smart Objects (e.g. a sensor that measures pressure) will be integrated via the Smart Object Integration component, which will communicate with the platform via the Message Routing component. The Process Designer provides the functionality to specify the manufacturing process in terms of abstract process models, which are to be realised by an ADVENTURE Virtual Factory, in a service-oriented fashion. The Process Designer will be embedded in a Dashboard along with all other required user interfaces.

To communicate with external systems, the Message Routing component's protocol and message format cannot be used, because of the diverse technologies of external systems.

Therefore, technical bridges will have to be implemented, which are represented by the Gateway components.

3 Functional Analysis

This chapter presents a system level analysis of the ADVENTURE platform from a functional point of view and shows how the components of the platform presented in section 2 will cooperate in order to provide the desired functionality to the end user.

Such system level functional analysis places each component in the overall context of the system and highlights its main interactions with the other components and with the user, through the graphical interface. Therefore, it provides a context for the detailed analysis and specification of the components that will be developed in section 5.

For the purpose of this functional analysis, two perspectives will be considered:

- **User Perspective,** showing how the components of the platform fit together in order to provide an intuitive and user-friendly interaction with the user at the ADVENTURE dashboard;
- **Process Execution Perspective**, showing how the components of the platform interact in order to instantiate, orchestrate, monitor, forecast and adapt virtual factory manufacturing processes.

3.1 User Perspective

This perspective deals with the interface that the system will provide to the end users through the dashboard. The ADVENTURE Dashboard will be the container for all components' user interface. It provides a common framework to implement the ADVENTURE graphical interfaces. Users will be able to access the whole ADVENTURE functionalities through the Dashboard in order to manage their Virtual Factories. The ADVENTURE Dashboard will be organized in four main areas:

- Process Design;
- Process Management;
- Partner Profile;
- Application Setup.

The Process Design area groups all the functionalities relevant for business process design, namely, process modeling, partner search and assignment, process simulation, process forecasting and process optimization.

The Process Management area groups all the functionalities relevant for business process execution and monitoring, namely process execution, process monitoring and process adaption.

The Partner Profile area groups all the functionalities relevant for Partner profile, services and products management, namely, profile creation and partner search as well as services and products description and search.

The Application setup area groups all the functionalities relevant for the platform and component specific configurations, namely users and templates management, smart object integration, gateways and transformation services configurations as well as alerts and notification rules setup.

The following table shows the mapping between the functionalities identified so far and the ADVENTURE Dashboard areas. This Dashboard arrangement forms the basis for the overall design of the ADVENTURE user interface.

The table shows the functionalities that were already identified in the DoW and the new functionalities that were identified during task T2.3 – User Requirements Analysis.

Table 1 - Functionalities to Dashboard Areas Mapping

ADVENTURE platform functionalities	Source	Dashboard Areas			
		Process Design	Process Management	Partner Profile	Application Setup
Virtual Factory Management					
Profile Management	DoW			X	
Smart Object Integration Configuration	D2.3				X
Gateway Configuration	D2.3				X
Transformation Services Configuration	D2.3				X
Message Routing Configuration	D2.3				X
Template management configuration	D2.3				X
Process Definition, Forecasting and Simulation					
Process Modeling	DoW	X			
Process Forecasting	DoW		X		
Process Simulation	DoW	X			
Partner Finding	DoW	X		X	
Process Optimization	DoW	X			
Real Time Monitoring and Process Adaption					
Real Time Monitoring	DoW		X		
Process Log	D2.3		X		
Alarm Rules Configuration	D2.3				X
Key Performance Indicators	D2.3		X		
Process Adaption	DoW		X		

The following subsections summarize use cases diagrams for each Dashboard area and show the component of the platform that supports each use case. The detailed description of each use case and the corresponding interface mockup will be presented later in Section 5.

3.1.1 Process Design Area

Process design encompasses a comprehensive set of use cases that support the set of activities involved in the design of the processes of the virtual factories. Its core component is the graphical process modeler, but it also includes tools allowing partner search and assignment, as well as process simulation and optimization. This means that the platform will provide to the user all the relevant tools for the design of the virtual factory within an integrated area in the Dashboard.

Section 4.1 details a use case scenario relating to process design that involves all the use cases presented in Figure 3. This scenario further develops the understanding of the system and shows how the different components of the platform cooperate in order to provide the desired functionalities to the virtual factory broker. This same scenario will be employed later, in WP7 to validate the technical solution that will be implemented in WP4, WP5 and WP6.

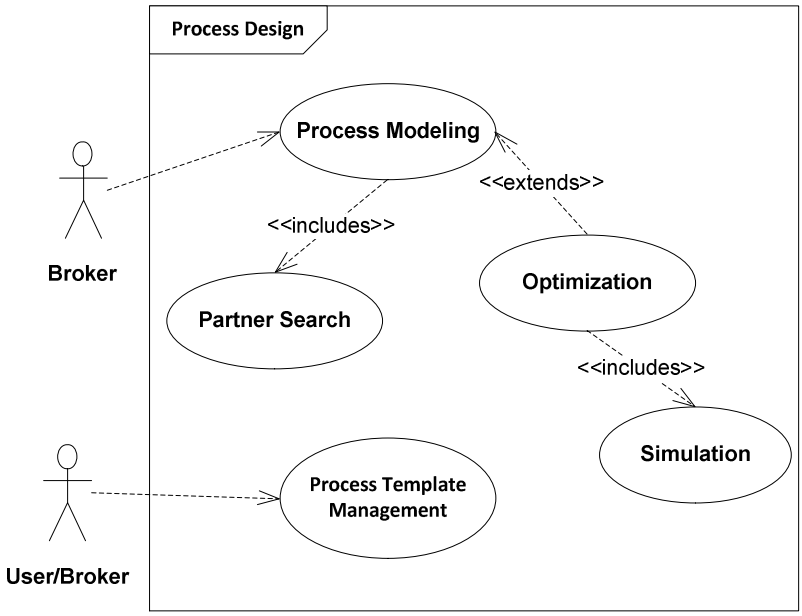

Figure 3 - Process Design Use Cases

3.1.2 Process Management Area

Process Management offers a comprehensive set of tools that fully support the creation, execution, monitoring and assessment of the manufacturing processes. This means that this environment will also provide tools for process forecasting and adaption that apply to running process instances, as shown in Figure 4.

Section 4.2 details a use case scenario relating to process management that involves all the use cases of process management area. This scenario further develops the understanding of the system and shows how the different components of the platform cooperate in order to provide the desired functionality to the virtual factory broker. It will also be employed later, to validate the technical solution that will be implemented.

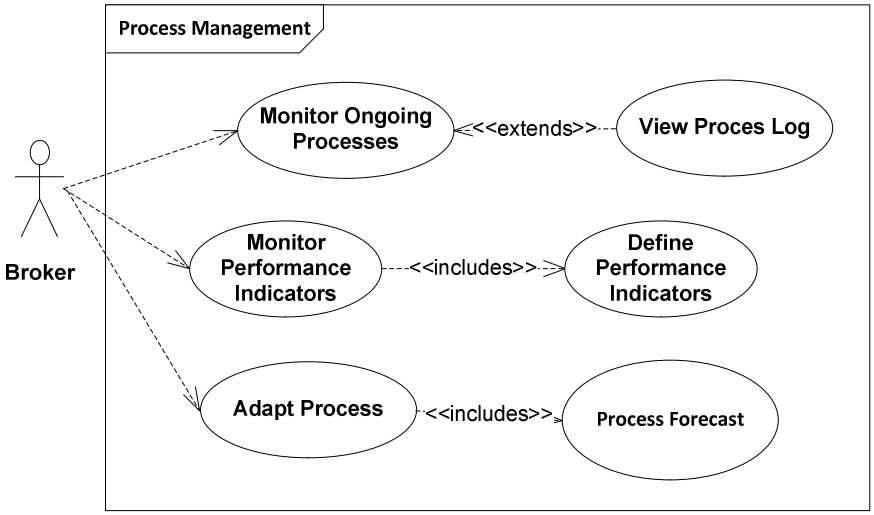

Figure 4 - Process Management Use Cases

3.1.3 Partners Profile

This Dashboard area provides the set of tools required to maintain the profiles of the ADVENTURE partners together with , tools for partner search based on semantic annotation. Profile management involves both the description of the business attributes of the partners (e.g., technical skills, delivery time, prices, location, certification, etc.), as well as the interoperability of ADVENTURE services implemented by the partners allowing their participation in ADVENTURE virtual factories.

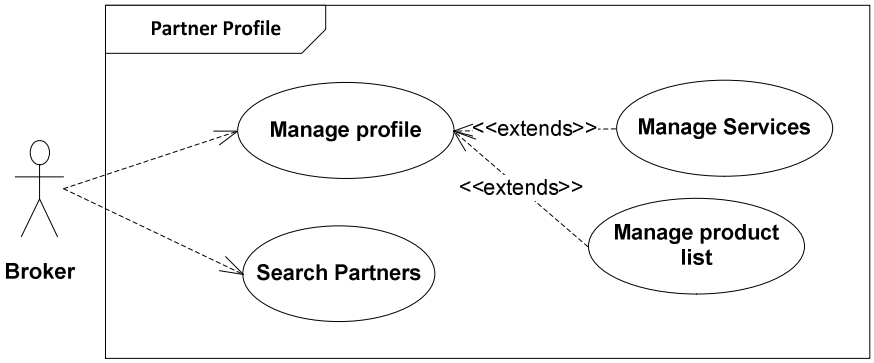

Figure 5 - Partner Profile Use Cases

3.1.4 Application Setup

Application Setup provides a comprehensive administration environment with all the configuration use cases, including the application level configuration (e.g., user management) and the component specific configuration (e.g., alerts and notification rules of the monitoring component).

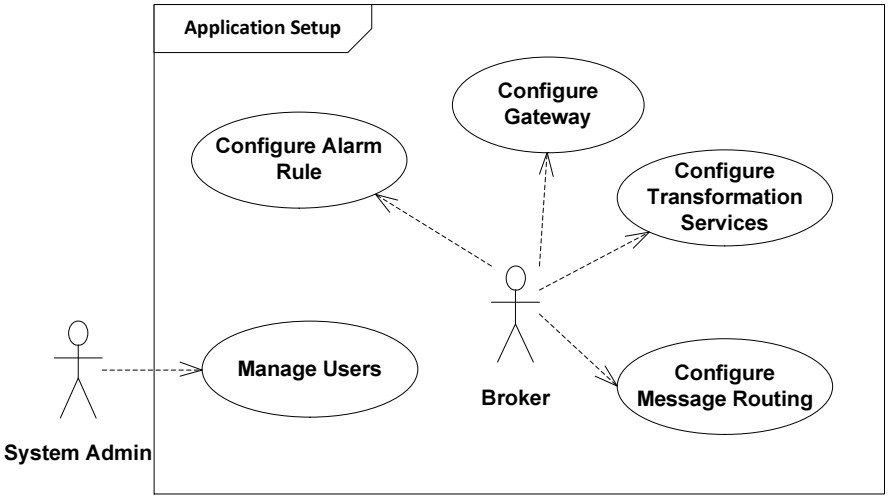

Figure 6 - Application Management Use Case Package

An important consideration at this point is the fact that the user may access the same functionality through different components. For example, the Data Provisioning and Discovery component will offer an autonomous interface for partner search but this functionality will also be available within the Process Design area, so that partners can be automatically searched and assigned to process steps based on criteria specified in the process model.

Similarly, the interface for the configuration of the Monitoring component will be included in the Application Setup area but, the Process management will also offer a limited and expedite configuration interface, for example, for alert rules.

3.2 Process Execution Perspective

This perspective deals with the execution of the virtual factories processes. Therefore, it provides the functionality required to run, orchestrate, forecast and dynamically adapt business processes. Figure 7 shows the main components involved in the execution of the smart processes. Hereafter, just a brief description of the main interactions between these components will be presented. A more detailed analysis will be provided in Section 5.8.

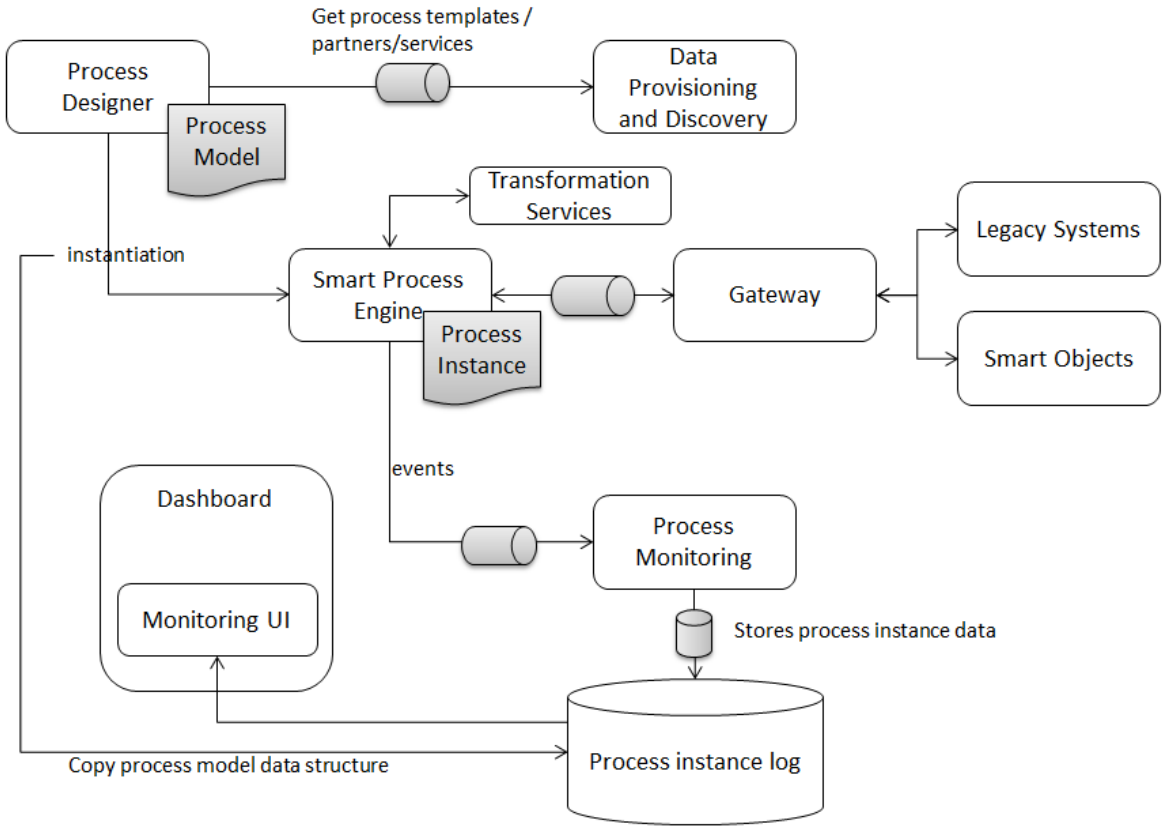

Figure 7 - Process Execution Overview

3.2.1 Process Execution

The core component of this layer is the Smart Process Engine (SPE). This component orchestrates all the invocation of services in the legacy systems of the virtual factory partners (e.g., manufacturing order placement and order status update). It also invokes the gateway services that collect data from the smart objects and other external systems. In Section 4, UML sequence diagrams provide detailed information on the interactions between the SPE and the Gateways.

Two main types of processes will be managed by the SPE component: macro-processes and micro-processes. The former correspond to long running processes involving multiple partners and probably lasting for long periods (e.g., weeks or even months). On the other hand, the latter will be much simpler processes that periodically collect data from the external systems, e.g., a smart object or variables directly read from legacy systems (e.g. stock level).

3.2.2 Events Notification

The SPE will notify the events occurring during the execution of the processes through the message broker. In order to present the appropriate information to the end user, both the real time process monitoring data and the process analytical data that will be presented at the dashboard by the will rely on the data stored in the cloud.

According to this loosely coupled architecture, the SPE component does not have a user interface and the monitoring component does not have any direct interactions with the gateways. In fact, the monitoring component receives all data relating to the execution of the process data through the events produced by the SPE.

3.2.3 Actions Upon the SPE

At run-time, the virtual factory broker may want to act upon a running process, e.g., stop or cancel the entire process, or just a particular step. Since the SPE does not have its own user interface, the monitoring component captures such user commands at the dashboard interface and acts upon the SPE accordingly.

Section 5 provides detailed information on the actions that the monitoring component may invoke on the SPE (e.g., stop/cancel a process/task).

3.2.4 Process Adaption

Process models are prepared in the Process designer. Each time a new process is initiated, the process model will be instantiated and the newly created process instance will be completely disconnected from the original process model, i.e., during runtime, if there is some adaptations or changes , they will not be reflected in the process model but only in the process instance. The SPE only “sees” the process instance data (which contain all the information that the SPE needs to run the process). Therefore, process instances can be changed during their execution (process adaptation), without any constraint imposed by the process model.

3.2.5 Message Broker

The platform will implement a loosely coupled architecture where all the interactions between the components will take place through a message broker. This component provides message buffering, synchronous and asynchronous message delivery, binary data transfer and communication to other message routing components that can work together in a server cluster in a final production system.

The interactions of the message broker with the other components of the platform are described in Section 5 using UML sequence diagrams.

4 Process Overview

In order to further develop the global understanding of the system, demonstrate its practical usefulness and verify the completeness of its functionality, two scenarios relating the main functionalities of the system will be discussed hereafter.

These scenarios will be further employed in work package 7 to validate the technical implementation of the system that will carry out in work package 6.

4.1 Process Design Scenario

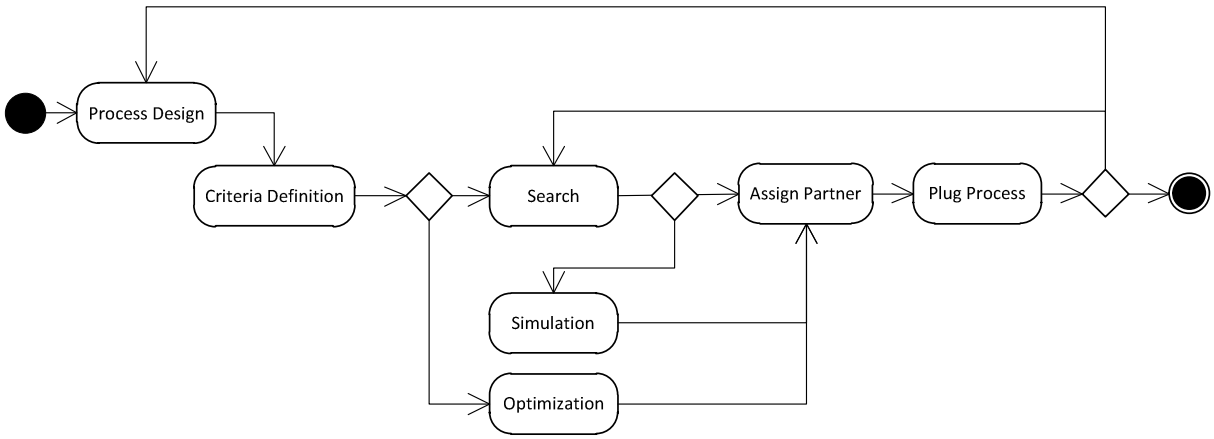

Figure 8 - Process Design Use Case Scenario Activity Diagram

In designing a virtual factory process, a typical sequence of steps is as follows:

- The user creates the process diagram using the graphical editor
- The user defines the partner selection criteria
- In order to select the partners, the user may:
    - simply search and select partners manually
    - invoke optimization component that will list the best set partners that fulfill the requirements previously specified
        - The Optimization component will invoke the simulation tool to assess and compare different combinations of partners that may be consider and provides a proposal to the user
        - The user may accept the proposal of the optimization tool or edit it
- After that, the user is able to do some simulations and changes manually and assign the definitive set of candidate partners to each step
- Once the set of candidate partners is identified, the user starts the plug process for each partner involved in the virtual factory
- The execution of the plug processes is monitored in the process management package
- If the plug process is successful the virtual factory will be ready for business operation

If one or more partners reject the work, the user goes back to the user selection step and looks for a new set of partners

Figure 8, shows the sequence of steps involved in process design. In order to enhance system usability all the use cases (except plug process monitoring) should be available at the same user interface.

4.2 Process Management

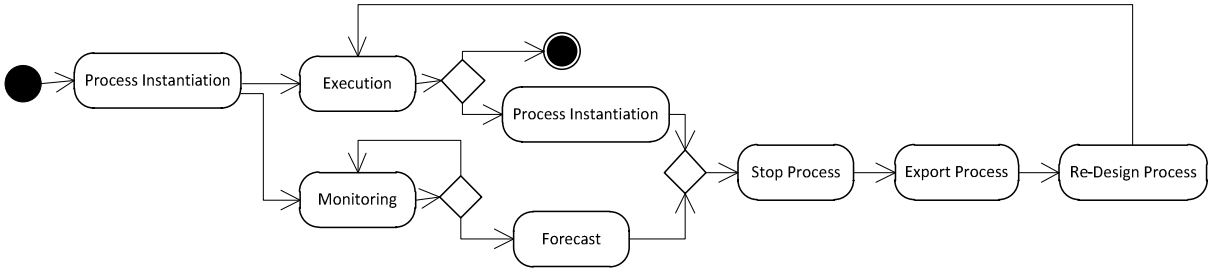

Figure 9 - Process Management Use Case Scenario Activity Diagram

In managing the processes of a virtual factory, a typical sequence of steps is as follows:

- A new process instance is created from the selected process model;
- Once a new instance is created the Smart Process Execution will orchestrate the execution of its steps. The Monitoring component will log the relevant events in the cloud and provide process status at the dashboard;
- When the Monitoring component detects a non-conformity in the execution of a process step (e.g., a delayed or defective delivery), an alert will be thrown at the Dashboard. In addition, each time the smart process engine component triggers an event, the monitoring component captures and processes this event;
- Then, the user may decide to invoke the Forecasting component in order to access the impact of that non-conformity (for example a late delivery) on the overall of the process – for example maybe the delayed delivery will have no impact since there is some flexibility in the timing or other parts have been delayed;
- If process forecasting shows a potential process level non-conformity, the user may decide to perform a process adaption;
- Assuming so, the user halts the execution of the process and exports the running instance to the process design environment;
- There the user updates the model of the process (which may involve, for example, the assignment of new partners or the a re-scheduling of the process steps);
- Then, the process instance managed by the SPE is updated and its execution restarted.

Figure 9, shows the sequence of steps involved in process design and the ADVENTURE tools that support each step. In order to enhance system usability all the use cases should be available at the same interface.

5 Component Level Functional Specification

This section of the document contains a subsection for each one of the components defined in *D3.1 - Global Architecture Definition* where the user interface functionalities and as well as functional interaction with other modules are described.

The subsection for each component is structured as follows:

- **Overall functional characterization.** This provides a short summary of the main functionalities to be implemented by the component;
- **Use cases.** This identifies and describes the use cases to be implemented by each component. In the use case diagram of a component, an actor may be a human interacting with ADVENTURE through the GUI, or another component of the platform that invokes a service. For the GUI use cases, the description includes an interface mockup and a detailed description of sequence of events;
- **Component interaction**: This point presents the internal structure of the component and describes in more detail its interactions with the other components of the platform;
- **Conceptual data model**. This presents a preliminary analysis of the data structure that will support the component;
- **Parameters to take into account in the technical specification**: A final table is shown for each component. This table contains generic and component specific criteria which will be acting as an input to T3.3 for guiding the selection of technologies. The tables show how important a specific aspect is for a component (e.g. the performance).

  The generic criteria that will be considered are the following:

  - **Maturity & Stability:** Selecting a stable and mature solution maybe one of the key criteria for all components of ADVENTURE for ensuring a usable implementation of the ADVENTURE. However, for other components using cutting edge technologies may be more relevant;
  - **Regularly Updated:** The ICT market is developing significantly faster than any other market in the world and technologies which haven't been updated within 3 years or more may be considered dead. As such, the consortium considers it important to choose as a base technologies that are updated regularly to reflect new market constraints;
  - **Technical Up-to-Datedness / Appeal:** Similar to the update policy, the technical up-to-datedness is considered as being important criteria. This will allow the consortium to use cutting-edge solutions and modern approaches for their development, hence creating more innovative solutions which will attract more attention during the project dissemination activities;
  - **Open Source:** Even if an existing technology may be used as a base for a component, it is very likely that the technology needs to be extended or changes in order to meet all project requirements. As such, open source solutions are likely to be preferred but in certain circumstances where this is not the focal RTD, or that this is not the market reality. off-the-shelf solutions can be of preference;
  - **Non-Infecting:** As specified un the CA, solutions that come with an infecting license such as GPL should be avoided and Apache 2 licenses should be preferred;

- **Code Quality:** A good code quality is required for any technology that is selected as a base for components;
- **Extensibility:** As a potential alternative to a good quality open source solution, a technology with well-defined extendibility may be considered, e.g. in terms of plugin mechanisms;
- **Community:** For clarifying questions and discussing possible problems of a technology, a strong community should be available. This community could also be considered to be represented by the members of the developing organization for providing free or commercial support;
- **Performance:** Performance is an important criterion for all core components of ADVENTURE although the meaning of "performance" might differ from component to component;
- **Reuse of existing developments:** Reusing existing solutions of the partners should be preferred as those solutions that have been developed by the consortium partners will most likely be easier to adopt;
- **EU project origin:** Bearing in mind the origin of ADVENTURE in the EU FP programs its leading-edge nature, solutions by other EU RTD projects can be quite useful in the context of a project although the juxtaposition is that most likely these technologies may not be mature or market accepted. However, in general this criterion does not have a high priority compared to others;
- **Platform (Portability):** If possible, technologies should be preferred that allow a deployment into different platforms. For server based components such as the storage component, this will basically mean support of Linux and Windows systems. For components such as the Dashboard this will mean a support of all major browsers and for protocols this will mean the existence of libraries for different mobile, web or desktop programming languages;
- **Open Standards Compliance:** Whenever an open standard exists, solutions should be preferred as long as those standards are openly accessible and applicable and as long as they are real world standards (defacto or dejure) that are actually used in the market and that do not hinder the technical up-to-datedness as described above;
- **Interoperability:** If possible, technical solutions should be preferred that provides a good base for achieving interoperability. For example, instead of a closed binary data format for querying or saving data in the cloud, an open and XML based solution should be preferred as it will potentially ease the interoperability with other components and thirds party systems. This criteria also reflects the integration capability with third party systems - e.g., ERPs

At the end of each component in this section 5, a summary table is given which at a high level identifies a set of parameters which are perceived to be important for that specific component – an example is given below. Some of the parameters (generic) are the same across all components although each parameter may still have a different importance to each component – for example Maturity. Other parameters are specific to each component – for example 'web based editor' is considered of interest for the process designer. The 'importance' measurement is on a scale of --- to +++ since it is not the intent, nor could it be, to provide an exact scientific formula for the section.

For example from the table below; regarding generic parameters; “Community”, it is often perceived to be important that a strong community exists in order to clarify questions and discuss possible problems of a technology. Regarding the specific parameters, an example, a balance has to be taken between e.g. “Scalability” which is not strictly necessary in the project but –may- be needed in a real world situation, However, since the projects focus is not on scalability research this has low importance; whereas using existing open projects may have high importance since it allows greater uptake.

This table is then reutilised in the technical specification where the potential technology sections are assessed and analysed against the parameters from the functional table and their importance. This then will provide conclusions on the direction of certain technologies that should be selected in the actual implementation.

Table 2 - Example of a Parameters Table for Process Design component (extract only)

Parameter	**Importance (- - - +/- +++)**
Generic Parameters	
Maturity & Stability	++
Regularly Updated	+
Technical Up-to-Datedness / Appeal	+
Open Source	+++
Non-Infecting	+++
Etc.	
Specific Parameters	
Scalability	+
Web based editor	+++
Etc.	

5.1 Dashboard

The Dashboard component encompasses all the user interfaces of the platform. This includes stand-alone use cases that are not component specific (e.g. user management, dashboard configuration and session management), as well as the component specific interfaces e.g. monitoring view, process designer, profile editor, etc.).

5.1.1 Overall Functional Characterization

Users will access the whole ADVENTURE functionality set through the Dashboard in order to manage their Virtual Factories.

As stated in section 3, the user interface is organized in four areas so that the Dashboard will contain one main section per area.

It also provides a login system to ensure secured access and a session management system to keep track of the access information. This dashboard is connected with Cloud Storage component.

This component will also provide a common framework to implement the ADVENTURE graphical interfaces.

The dashboard will be organized in 3 main levels: Level 0 corresponds to the Homepage and menu structure that permits the access to all functionalities. Level 1 is the entry page for each of the four areas described in section 3. Level 2 is component specific and will be developed by each component team and included in the dashboard as a portlet.

5.1.2 Use Cases

Figure 10 shows the level 0 and level 1 use cases. Level 2 use cases will be presented at the corresponding component subsection.

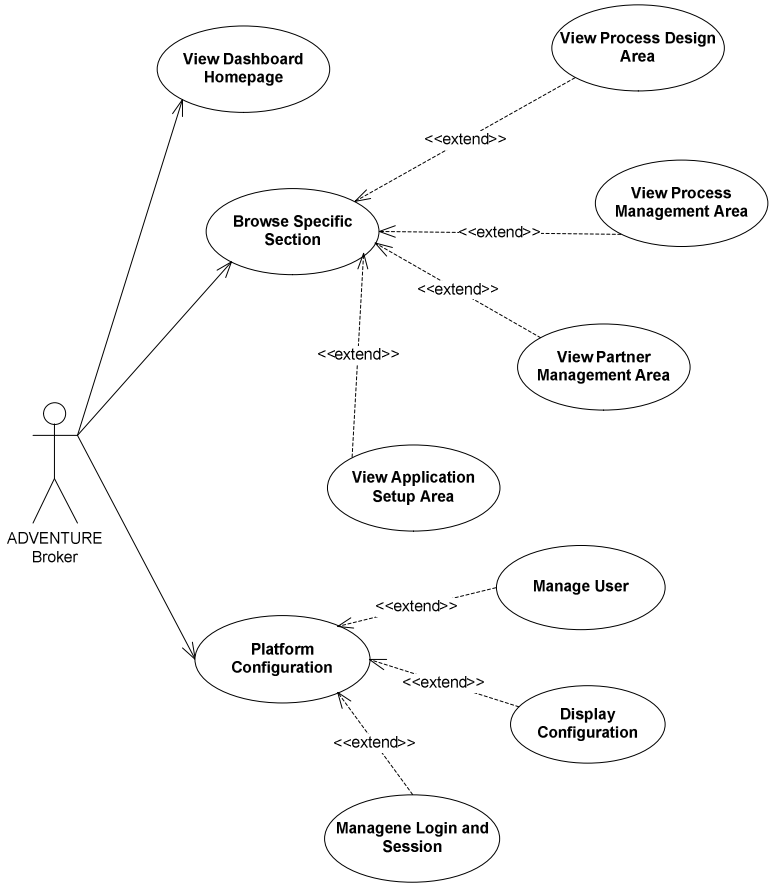

Figure 10 - Dashboard Main Sections Use Case Diagram

5.1.2.1 View Dashboard Homepage

The Dashboard UI is organized in four areas: process design, process management, partner management and application set up so that the dashboard will contain one main section per use case package.

Figure 11 presents an 'Overview of ADVENTURE Home Panel' respectively and it is observed that user can visualize the overall high level status information of each of the processes. This visualization process provides the user to monitor and control the collaborative activities as needed to run the virtual enterprise successfully.

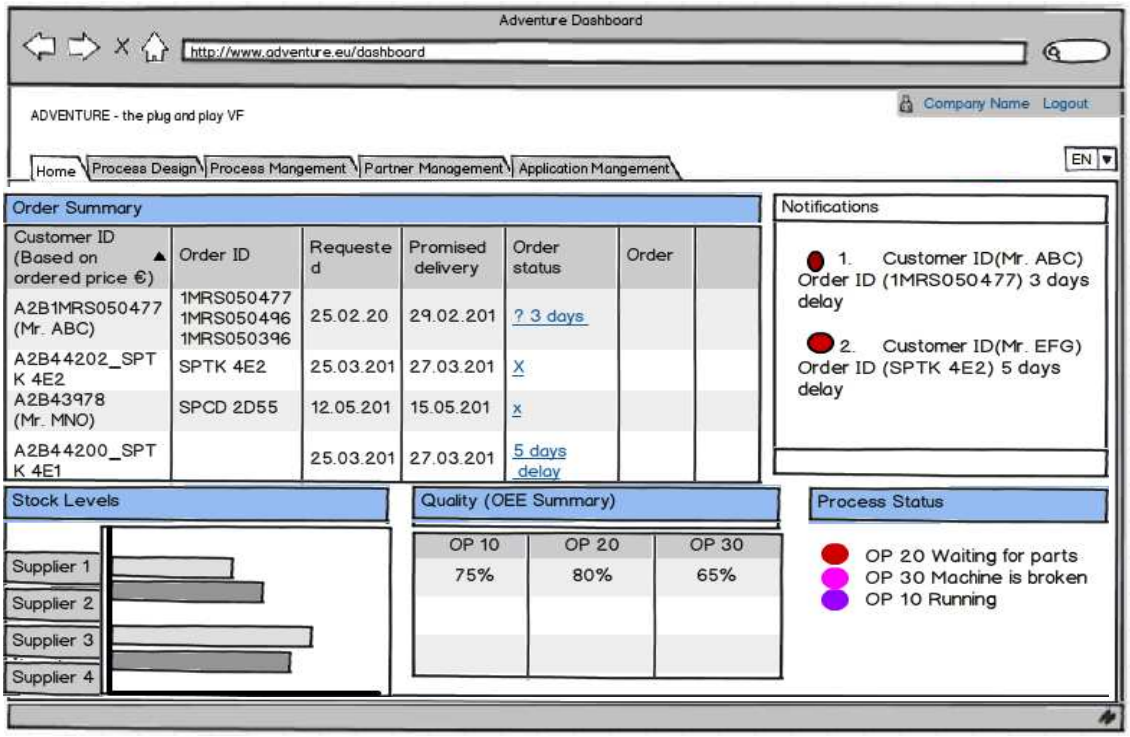

Figure 11 - Overview of the Dashboard Homepage (Level 0)

Use Case D-U01	**Name: View Dashboard Homepage**
Description	Broker can visualizes the dashboard homepage
Actors	ADVENTURE Broker
Pre-conditions	• Broker is logged in
Inputs	N/A

Events Sequence	• Broker can visualize the contents of the dashboard overview. In this dashboard overview broker can find important information to run the virtual enterprise. There are five sections in the dashboard overview: Order Summary, Stock Levels, Quality (OEE Summary), Notifications and Process Status.
	  • In the 'Order Summary' section, it shows all the customers IDs, and different orders status. Broker can select specific customer in order to display more detail information about each order.
	  • The 'Stock Levels' are visualized according to different suppliers. Broker can get brief stock levels of each of the suppliers' sites.
	  • In the 'Notification', the broker can check the warning message about the virtual enterprise. For instance, if there is a delay in any processes.
	  • The 'Process Status' shows the real-time situation of each step in the process. Different colours illustrate different statuses. For instance, red for risk, yellow for warning and green for running, etc. Clicking on any specific process will show up more details about the process status.
	  • In the 'Quality (OEE Summary)', broker can see the overall equipment efficiency (OEE).

Extension Points	• N/A

Outputs	• More detail information about order, stock, quality, environment, processes and notifications.
Post-condition	N/A
Requirements map	F4, F30, F37, F38, F37, F38, F29, F32
Related Mock up	Figure 11 - Overview of the Dashboard Homepage (Level 0)

5.1.2.2 View Process Design Environment

This Environment offers a comprehensive set of tools that support the set of activities involved in the design of the processes of the virtual factories. Its core component is the graphical process designer, but it also includes tools allowing partner search and assignment, as well as process simulation and optimization. The following figure shows the main functionalities that user is able to access within the process design tab.

Use Case D-U03	**View Process Design Environment**
Description	Broker and Administrator can visualize the process design environment
Actors	ADVENTURE Broker, ADVENTURE Administrator
Pre-conditions	• Broker/Administrator is logged in
Inputs	N/A

Events Sequence	• Administrator invokes Process Design functionality by selecting on the menu bar in the dashboard
	• Administrator can manage the Process Template created before by selecting a specific Process Template
	  • Administrator edits the Template
	  • Administrator delete the Template
	  • Administrator save the Template
	• Broker invokes Process Design functionality by selecting on the menu bar in the dashboard
	• Broker selects Process Modelling and then search new partner for the each step in the process
	• The system checks whether the process should be optimized. The system performs process simulation

Extension Points	• Use Case PO-U01 “Specify process model” • Use Case PD-U01 “Create New Process Model”
Outputs	• The process design environment is shown to the broker/administrator, where they can access the process manager, process designer, including partner search, simulation and optimization functionalities.
Post-condition	N/A
Requirements map	F4,F30,F37, F38,F37,F38,F29, F32
Related Mock up	Figure 12 - Process Design Environment Tab (Level 1)

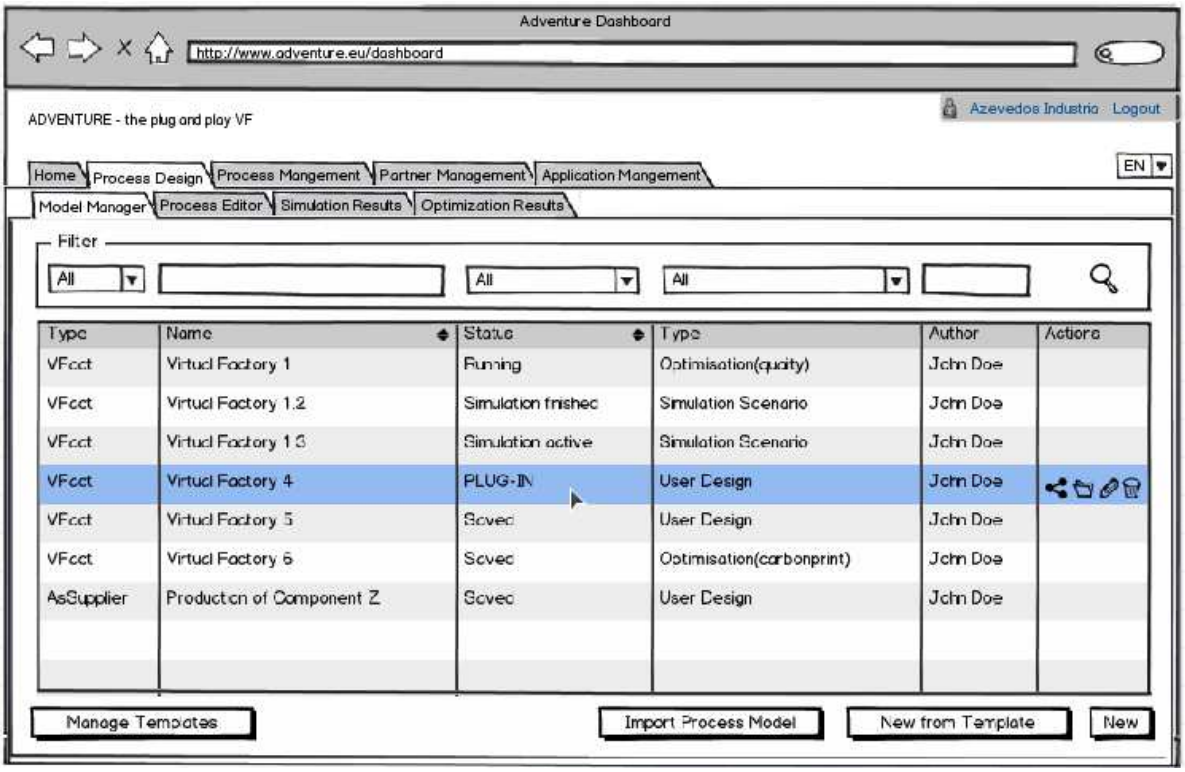

Figure 12 - Process Design Environment Tab (Level 1)

5.1.2.3 View Process Management Environment

The Process Management Tab offers a set of tools that fully support the execution, monitoring and assessment of the business processes. This means that this environment will also provide tools for process adaption that apply to running process instances. At the Alerts, the user can check the warning message about the virtual factories. For instance, when there is a delay in any process. User can also click an alarm for more details of any specific notifications.

The following figure shows the main functionalities that user is able to access within this tab.

Use Case D-U04	View Process Management Environment
Description	Broker can visualize the process management environment
Actors	ADVENTURE Broker
Pre-conditions	• Broker is logged in
Inputs	N/A
Events Sequence	• Broker invokes Process Management functionality by selecting the menu bar on the Dashboard • Broker monitor all on-going process • Broker can select one specific process to view the Process Log • Broker monitor Performance Indicator • Broker can define new Performance Indicator

	• Broker can manage Performance Indicator • Broker adapt process • System performs Process Forecast
Extension Points	• Use Case PA-U01 • Use Case PF-U01
Outputs	• The process management environment is shown to the brokers where they can access the process monitoring, process log, process adaption functionalities.
Post-condition	N/A
Requirements map	F4,F30,F37, F38,F37,F38,F29, F32
Related Mock up	Figure 13 - Process Management Tab (Level 1)

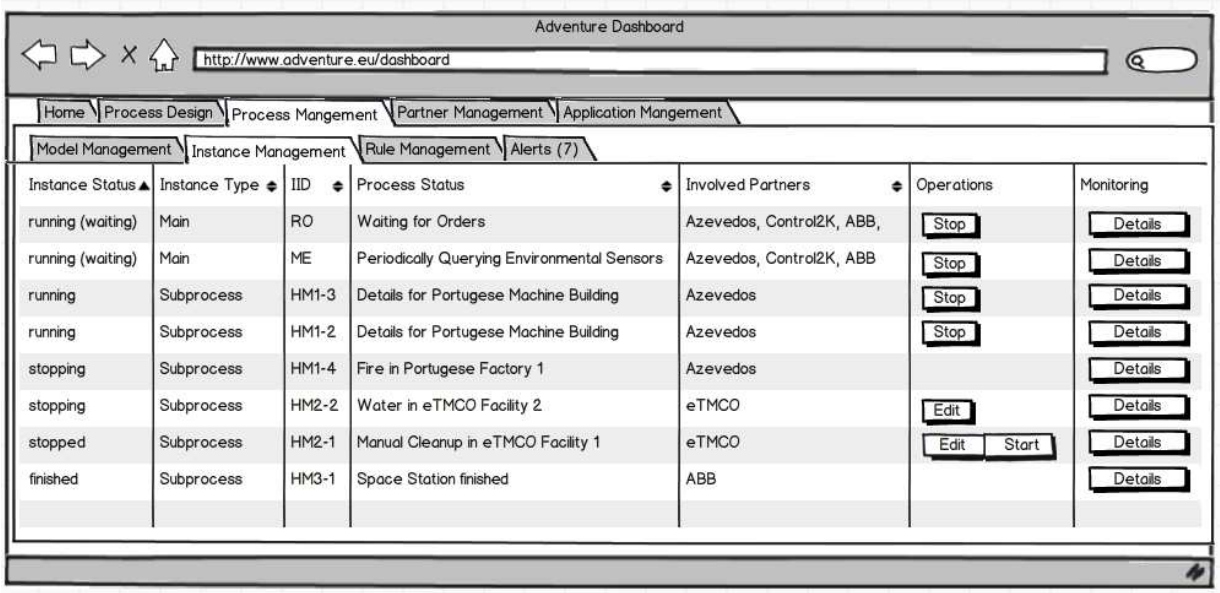

Figure 13 - Process Management Tab (Level 1)

5.1.2.4 View Partner Profile Environment

This tab gathers the set of tools required to maintain the profiles of the ADVENTURE partners and tools for partners search based on semantic annotation. Profile management involves both the description of the business attributes of the partners (e.g., technical skills, location, certification, etc.), as well as the interoperability ADVENTURE services they implement and that allow their participation in the ADVENTURE virtual factories. The following figure shows the main functionalities that user is able to access within this tab.

Use Case D-U05	View Partner Profile Environment
Description	Broker can visualize the partner management environment
Actors	ADVENTURE Broker
Pre-conditions	• Broker is logged in
Inputs	N/A

Events Sequence	• Broker invokes partner profile functionality by selecting on the menu bar in the dashboard
	• Broker can manage Broker Member profile
	  • Broker can manage the Broker Member services
	  • Broker can manage the Broker Member products
	• Broker can search Partners

Extension Points	• N/A
Outputs	• The partner management environment is shown to the brokers, where they can access the profile editor, partner finding, partner analytics functionalities.
Post-condition	N/A
Requirements map	F4,F30,F37, F38,F37,F38,F29, F32
Related Mock up(s)	Figure 14 - Partner Profile Tab (Level 1)

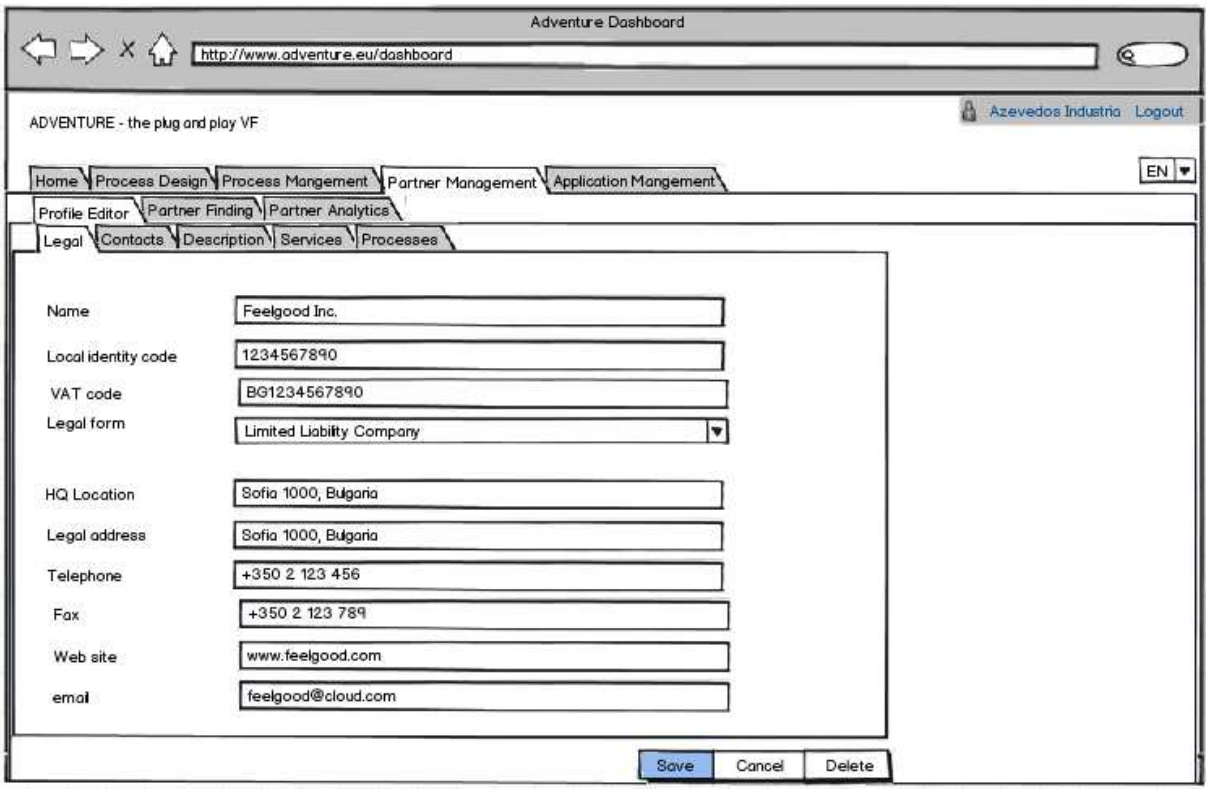

Figure 14 - Partner Profile Tab (Level 1)

5.1.2.5 Display Application Setup Environment

The Application Configuration tab gathers all the configuration use cases, i.e., the application level configurations, e.g., user management, and the component specific configuration, e.g., alert and alarm rules of the monitoring component and configuration of the gateways and the transformation services. The following figure shows the main functionalities that user is able to access within this tab.

Use Case D-U06	View Application Setup Environment
Description	Broker/ Administrator can visualize the application management environment
Actors	ADVENTURE Broker, ADVENTURE Administrator
Pre-conditions	• Broker/Administrator is logged in
Inputs	N/A
Events Sequence	• Administrator manages user • Administrator can create users • Administrator can create roles • Broker configures application • Broker configures alarm rule • Broker configures gateway • Broker configures transformation services
Extension Points	• N/A
Outputs	• The application management is shown to the broker/administrator where they can access the user management, display configuration, gateways configuration, alarms configuration, alarms configuration functionalities.
Post-condition	• Use case D-U07 “Broker already registered” • Use case D-U08 “Broker logged in” • Use case D-U01 “Dashboard overview is displayed”
Requirements map	F4,F30,F37, F38,F37,F38,F29, F32
Related Mock up(s)	Figure 15 - Application Setup Tab (Level 1)

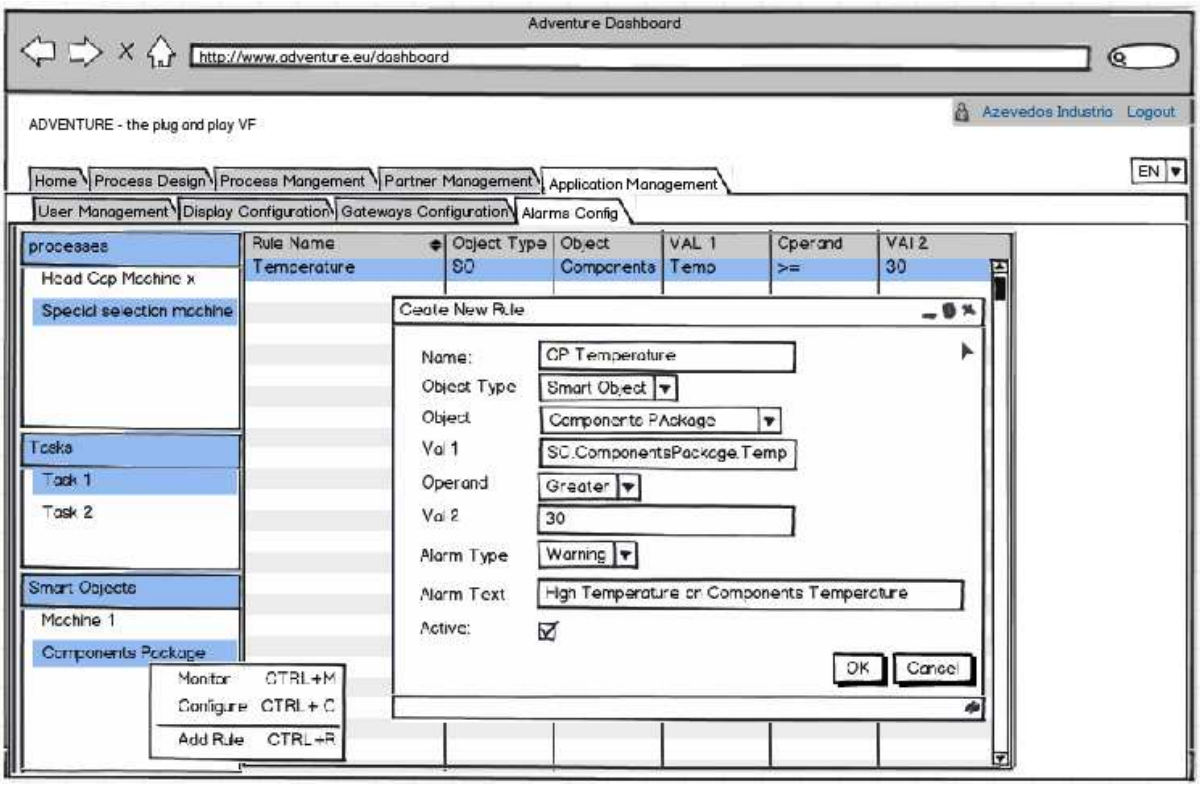

Figure 15 - Application Setup Tab (Level 1)

5.1.2.6 Configure Dashboard

The ADVENTURE dashboard can be configured according to user's own preferences such as adding a new widget, enlarging a specific widget, delete widget, etc. The Dashboard configuration use case diagram, use case specification and mockup are presented in Figure 16. It is seen that each of the process windows can be customized by using the symbols as presented top right hand corner of each process window.

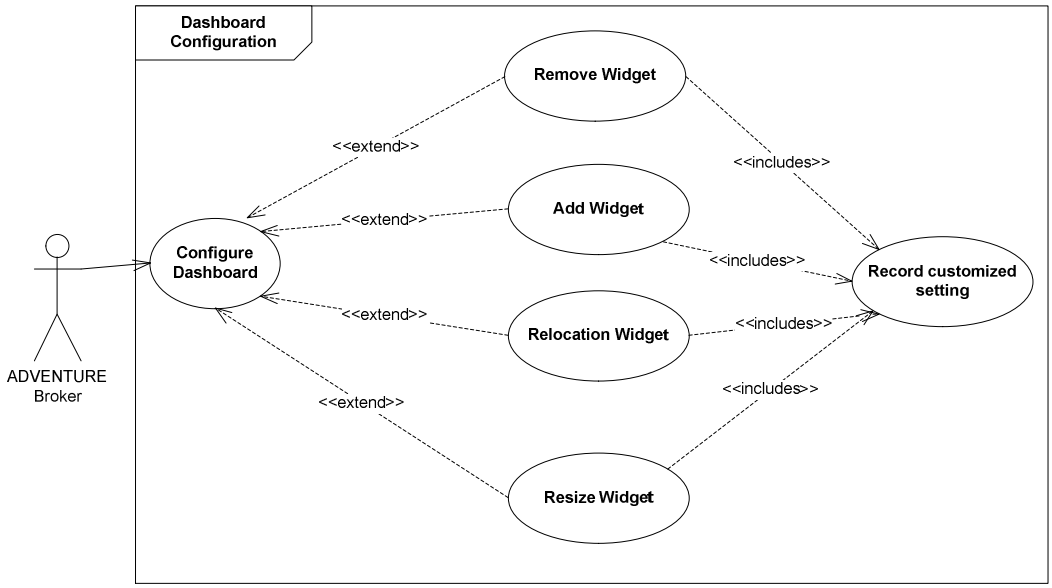

Figure 16 - Dashboard Configuration Use Case Diagram

Use Case D-U02	**Name: Configure Dashboard**
Description	The configurable dashboard enables broker's to customize the dashboard overview according to needs.
Actors	ADVENTURE Broker
Pre-conditions	• Broker is logged in
Inputs	N/A

Events Sequence	• Broker invokes the configurable Dashboard screen
	• The Dashboard Home screen is ready to be customized
	• Click on the setting button, broker has the following options to customize the dashboard:
	  ◦ Broker can remove the widget by selecting the "Delete" icon
	  ◦ Broker can add other widgets by selecting "Add", and then a list of all the dashboard components is displayed.
	  ◦ Broker can move the location of widget by drag and drop.
	  ◦ Broker can resize the widgets by selecting the resizing button.
	• Broker save the setting and the customized setting will record in the cloud storage for future use.

Extension Points	• Use case D-U01
Outputs	• Individual template of the dashboard is ready to use
Post-condition	N/A
Requirements map	F65
Related Mock up	Figure 17 - Display of Dashboard Configuration Process (Level 0)

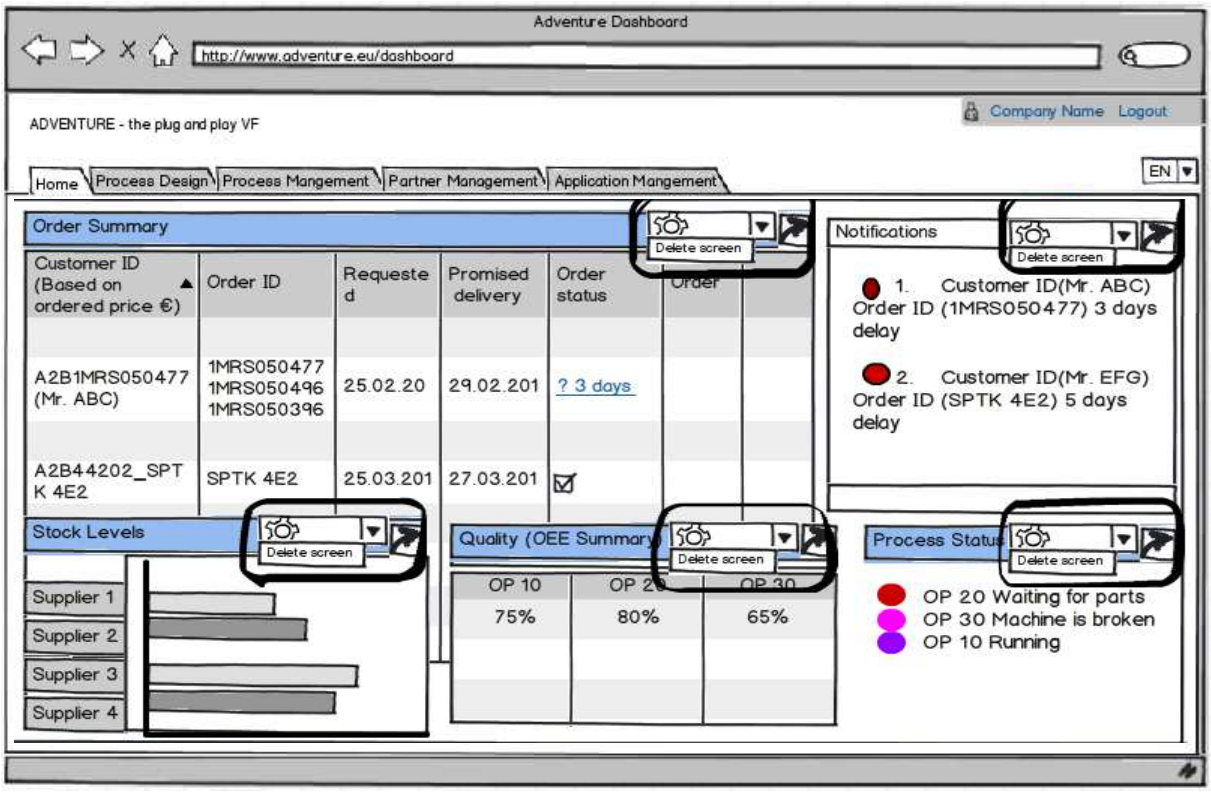

Figure 17 - Display of Dashboard Configuration Process (Level 0)

5.1.2.7 User Registration

The system will be secured by a user authentication system provided by the framework with which the dashboard is implemented. ADVENTURE's user, which can be a company or many users within a company need to be logged in before accessing data in the private area of the system. In order to access into the dashboard functionalities, users need to register beforehand in the ADVENTURE system. Successful registration process only allows the potential users to login the platform through secured user name and password. This registration process is divided into use case diagram, use case specification and mock ups. The use case and mock ups are presented in Figure 18 and Figure 19 respectively.

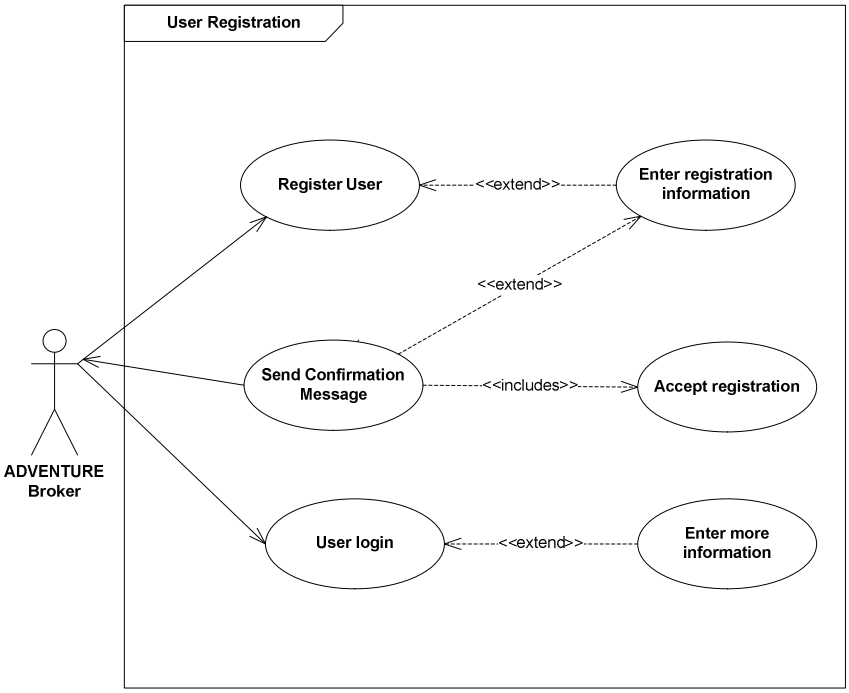

Figure 18 - User Registration Use Case Diagram

Use Case D-U07	User Registration
Description	Broker must register with the web site before they are able to participate in ADVENTURE
Actors	ADVENTURE Broker
Pre-conditions	• Broker is not a member of ADVENTURE.
Inputs	N/A

Events Sequence	• Broker navigates to the homepage and selects the “ADVENTURE Online Registration” link.
	• Dashboard displays a new user registration form to be populated accordingly.
	• Broker enters all fields as required for registration process.
	• Broker selects “SEND FORM”.
	• System responds with visual clue indicating the processing of User Registration.
	  • System validates User Information.
	  • System validates Email Address.
	  • System sends message to new user’s E-mail, confirming preliminary registration. The message includes with user name, user password and a confirmation link.
	• Broker clicks the confirmation link and login the system with the user name and password as provided.
	• System presents welcome page for the new user.
	• Broker can choose whether they like to join the virtual enterprise

	community or not. If the answer is yes, broker is requested to provide more information about his/her company.
Extension Points	• N/A
Outputs	• Be able to login the system with new user ID and password.
Post-condition	• Broker might need to provide more information after preliminary registration. This information can be used to facilitate user to join into the virtual enterprise community.
Requirements map	N/A
Related Mock up	Figure 19 - Display of ADVENTURE User Registration Process (Level 0)

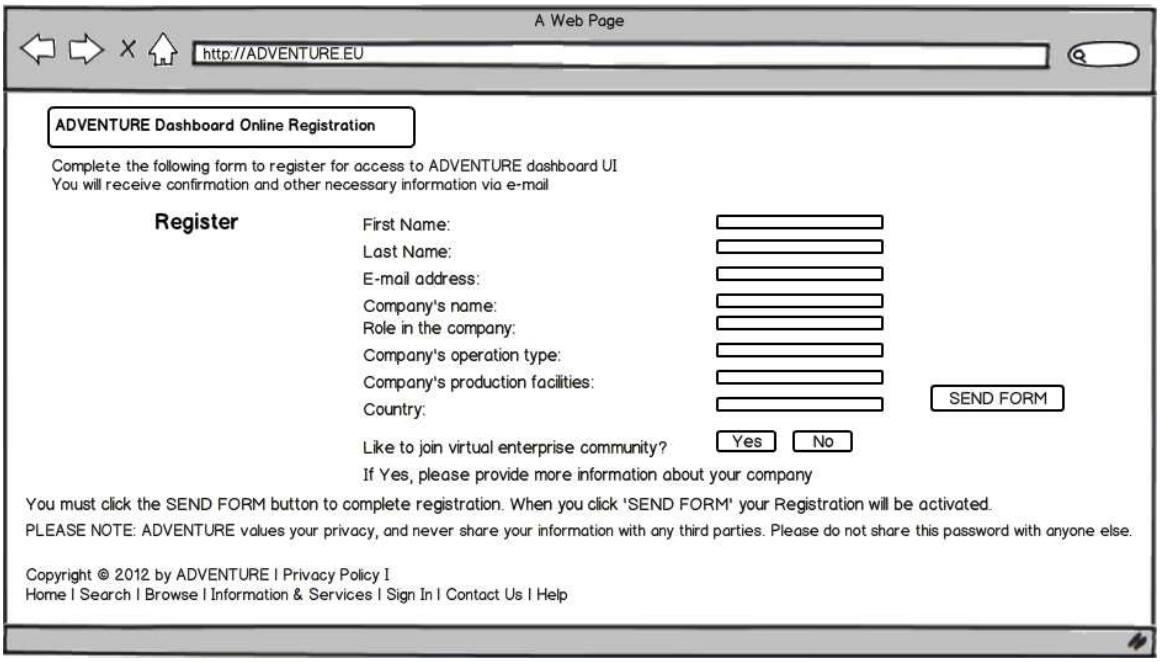

Figure 19 - Display of ADVENTURE User Registration Process (Level 0)

5.1.2.8 Login and Session Management

The user login and session management processes can be divided into use case diagram, use case specification and mock ups. Figure 20 display the use case and mock ups for user login and session management respectively. This login functionality should have to be defined according to user's type (broker, customer and supplier) as displayed in Figure 21. The system needs to be signed out for ensuring data security or data encryption, and the session is ended. Before ending the session, the actions performed by the user are stored within the cloud storage.

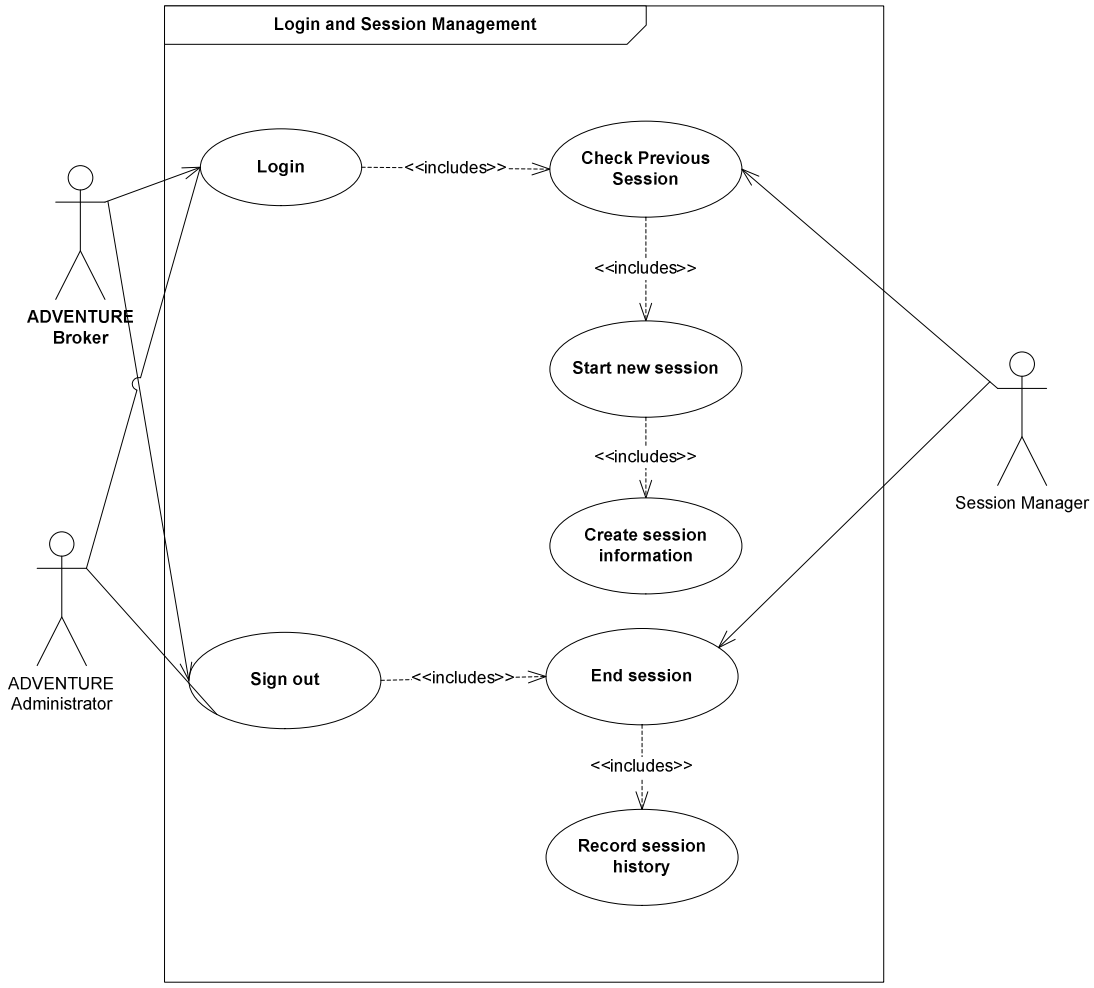

Figure 20 - Login and Session Management Use Case Diagram

Use Case D-U08	Login and Session Management
Description	Keep track of broker/administrator activity and record the session history
Actors	ADVENTURE Broker, ADVENTURE Administrator, Session Manager
Pre-conditions	• Broker is not logged in.
Inputs	N/A
Events sequence	• Broker/Administrator selects “Login”. Broker/Administrator can select different roles and login the system with correct username and password accordingly. • Broker/Administrator can also select the function ‘Save the password’. The dashboard overview screen will be displayed (Figure 20). • The session manager checks whether a previous session existing for the given Broker/Administrator or not. If not, then create a new session. • Relevant session information will be created by Session Manager,

	i.e. session identifier (session ID) and other session data (user name, account number, etc.) and tracking information (which applications /components are open and using, which documents each application has opened) • Broker/Administrator sign out the system. • The session manager evokes to record the session information in the cloud storage. This can create session history for future use.
Extension Points	• Use case D-U07 “Broker/Administrator is registered”
Outputs	• Display more detail information about order, stock, quality, environment, processes and notifications.
Post-condition	N/A
Requirements map	N/A
Related mockup(s)	Figure 21 - User Login Process (Level 0)

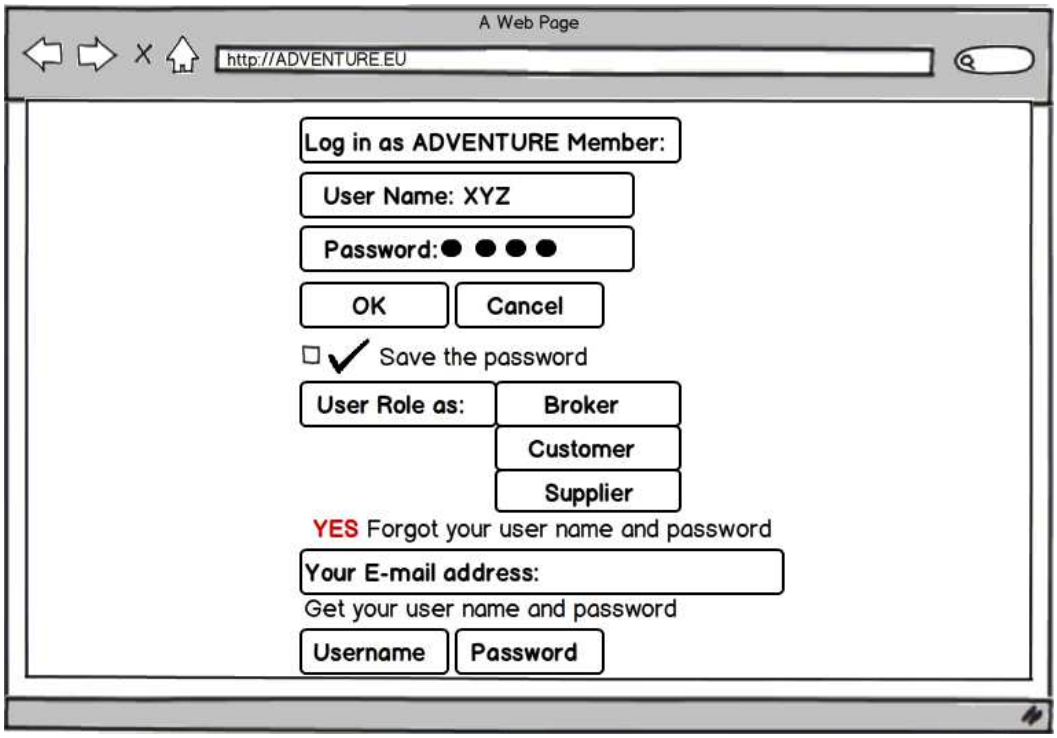

Figure 21 - User Login Process (Level 0)

5.1.3 Component Interaction

The Dashboard interacts with different components or modules as displayed in Figure 22.

From Figure 22, it is observed that user requirements are directly feed into the Application Server of the Dashboard. This Server contains the Dashboard Component Container, which is composed of Graphics Framework and ADVENTURE Component UI, User Management, Session Management and Template Management components.

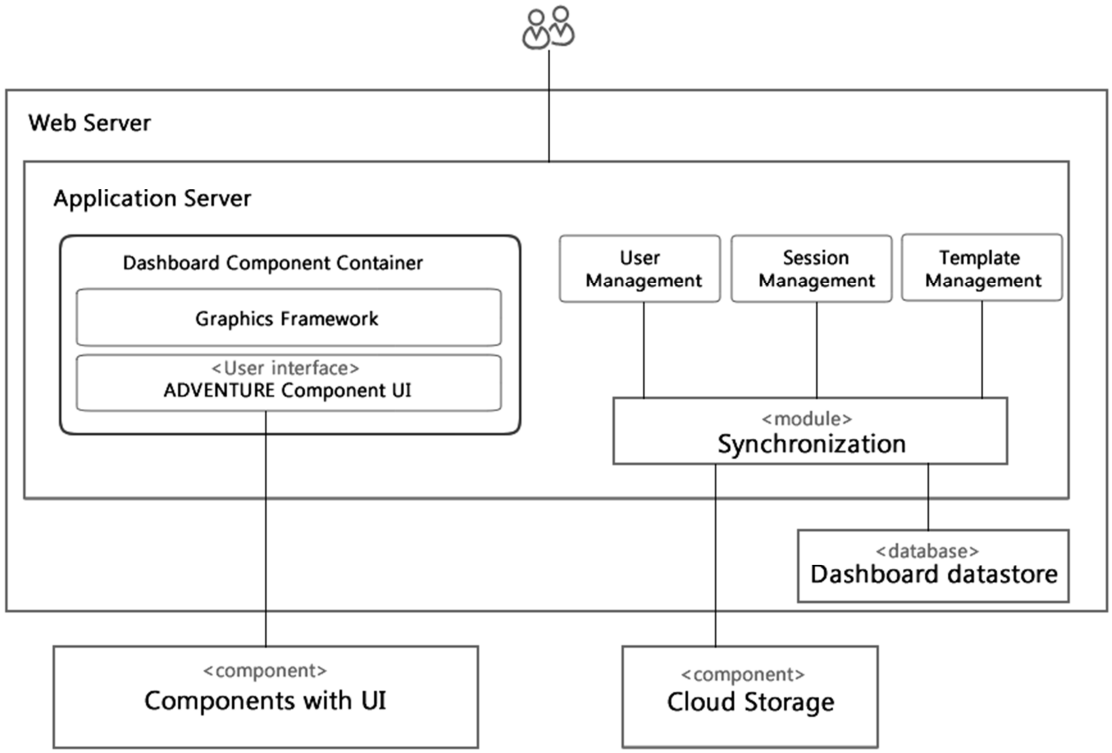

Figure 22 - Display of ADVENTURE Dashboard Architecture

The Dashboard Component Container interfaces with the Components with UI component according to the user requirement. The User Management component within the Application Server manages the required entry point to the dash board platform through logged in/off functionality. The Session Management component is responsible for managing the dash board session in terms of tracking an individual session with user ID. The Template Management component functions according to the user point of view, which allows the user to define a role and accompanying rights accordingly. These three management components (User, Session and Template) interact with the Cloud Storage and Dashboard data store components through Synchronization module as depicted in Figure 22. It can be mentioned here that dashboard component will need its own database for the requirement of its application server. This database type can be varied according to the chosen platform over which dashboard will be functioning. The cloud storage maintains necessary data or information as received from the user, session and template management modules.

5.1.4 Conceptual Data Model

This subsection will define the data entities for Dashboard components. This data model is composed of constructs such as Entity Name, Entity Relationship and Entity Descriptions. As explained earlier, the Dashboard component is considered as the entry point for all the users in to the ADVENTURE environment. It includes Dashboard Components Object, User Object, and Session Object as displayed in Figure 23.

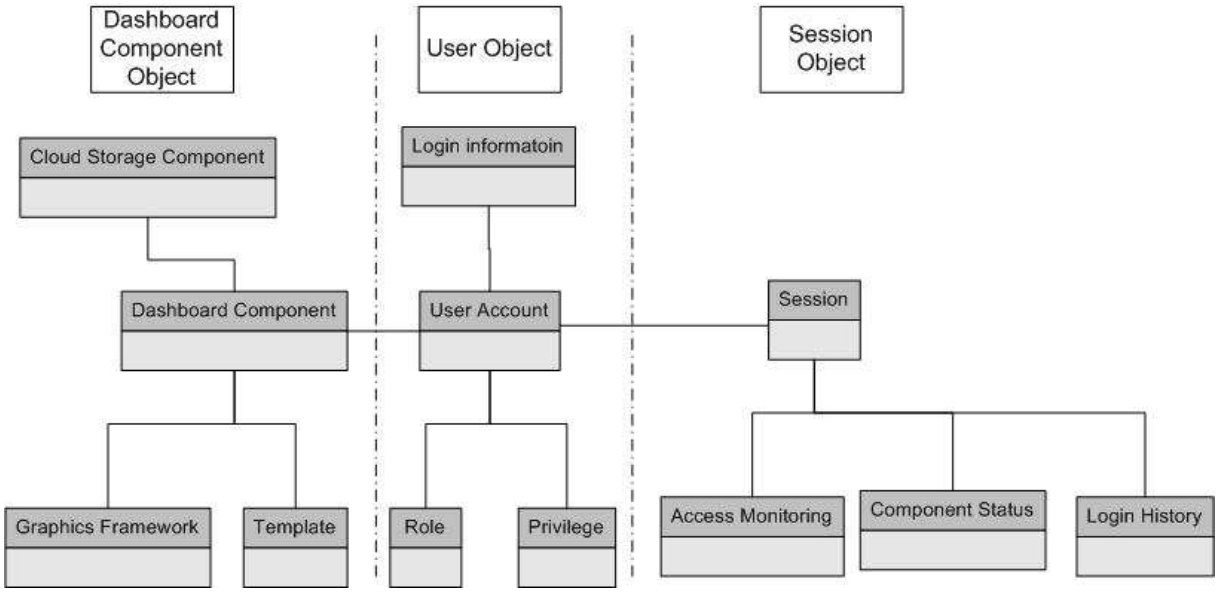

Figure 23 - Domain Model for ADVENTURE Dashboard

- Dashboard Component Object is designed to implement the configurable ADVENTURE Dashboard. By using this Dashboard, users can customize the Dashboard according to their needs, and their different roles or privileges. Dashboard Component Object includes, Dashboard Component, Dashboard Template and Graphics Framework.
- User Object provides users to control the users' accounts. Different users have different roles and different authorities, so they can manage the ADVENTURE processes and data, based on their roles and privileges. Within the User Object, Login information, User Account, User Roles and Privileges are managed.
- Session Object is used to maintain the session history between users and ADVENTURE platform. It includes access monitoring, component status and login history.

5.1.5 Parameters to Take into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	++
Regularly Updated	--
Technical Up-to-Datedness / Appeal	+/-
Open Source	-
Non-Infecting	-
Code-Quality	+
Extensibility	+++
Community	+
Performance	++
Reuse of existing developments	+
EU project origin	-
Platform (Portability)	+
Open Standards Compliance	-
Interoperability	++
Specific Parameters	
User friendly	+++
Visualization quality	+++
Browser Support	+++
Community Management	+++
Interface Protocol	+++
Authentication	+++

5.2 Process Designer

The Process Designer module is the key tool in the stage of designing Virtual Factories, which collaborates with assistive tools like simulation and optimization, to deliver ADVENTURE brokers with the core functionality to accomplish this task.

5.2.1 Overall Functional Characterization

The module's main object is the Smart Process Model and it provides the required functionality to manage its lifecycle and perform tasks related to the different phases, as described on *Figure 24*.

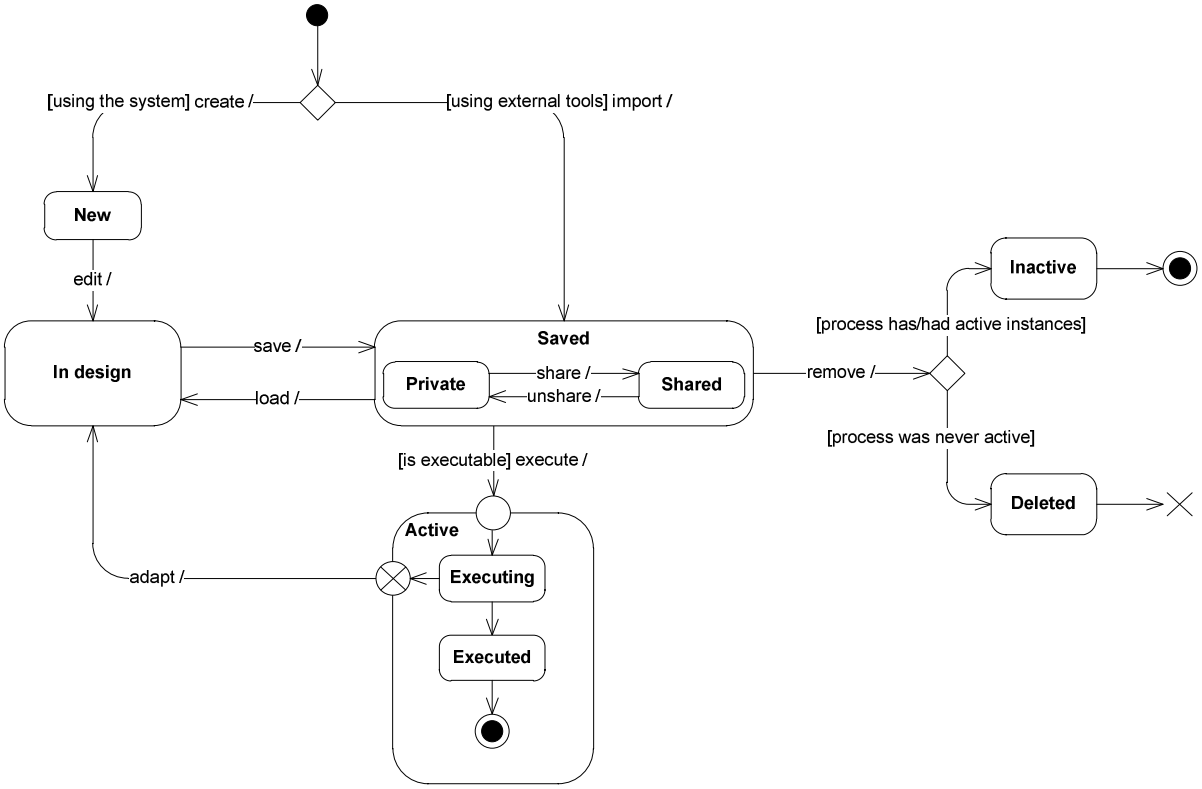

Figure 24 - Process Model Lifecycle States

A *new* Process Model is created in the Process Designer either as an empty model or based on a ready-to-use template from templates repository. As the ADVENTURE broker starts to design (edit) the process, the model enters its '*in design'* phase. This is the core phase for the Process Designer and the designer can perform several types of tasks in it:

- Manage process metadata with the goal to enable automation and make the process discoverable
- Design process models, e.g. add/configure/remove process activities or other process model elements, using the notation and semantics supported by the Process model design tool

- Use the simulation module to trigger simulation of the process and verify its qualities before executing it and potentially return to redesign for better results
- Use the optimization module to trigger optimization of the process for optimal business goal results and potentially return to redesign for better results
- Save for further work or share the designed process model
- Load, if the process model has been saved earlier or has been shared by another user

A process model that is *in design* state can be *saved*. Process models that have been designed in external tools, compatible with the ADVENTURE system, may be imported into the system and saved for further use too. Further, the process models can be exported in a standard format. *Saved* process models can be loaded back into *in design* state for further changes. By default, *saved* process models are *private* to their owners. However they can also be *shared* by their owners with other ADVENTURE members for exploration and collaboration. Those that have been *shared* can be unshared to become *private* again.

When a process model design is ready to be executed, e.g. process activities have services bound to them, a broker can request a new instance of the model to be executed by the Process Execution Engine. This changes the state of the process model to *active*. During their *active* phase, process models can eventually be in *executing* or *executed* state. These two states depend on the process model instances that are executed in the Process Execution Engine. If there's at least one model instance that is currently in execution state in the Process Execution Engine, then the model's state is *executing*. If there was at least one model instance that had been executed previously but none are executing currently, the model state is *executed*.

The *executing* and *executed* states control the last stage in the process model lifecycle. Models, which have at least one instance in *executing* state cannot be removed (become *deleted* or *inactive*). Process models that have instances only in *executed* state (past executions have occurred, but none currently) can be become *inactive*, but not *deleted* (physically removed), in order to preserve history of relations between processes, member profiles and services. Process models that have never been *executed* (no associated instances in *active* state exist) can be safely *deleted*.

Active processes that are *executing* can be opened in design when changes are needed due to required adaptations.

5.2.2 Use Cases

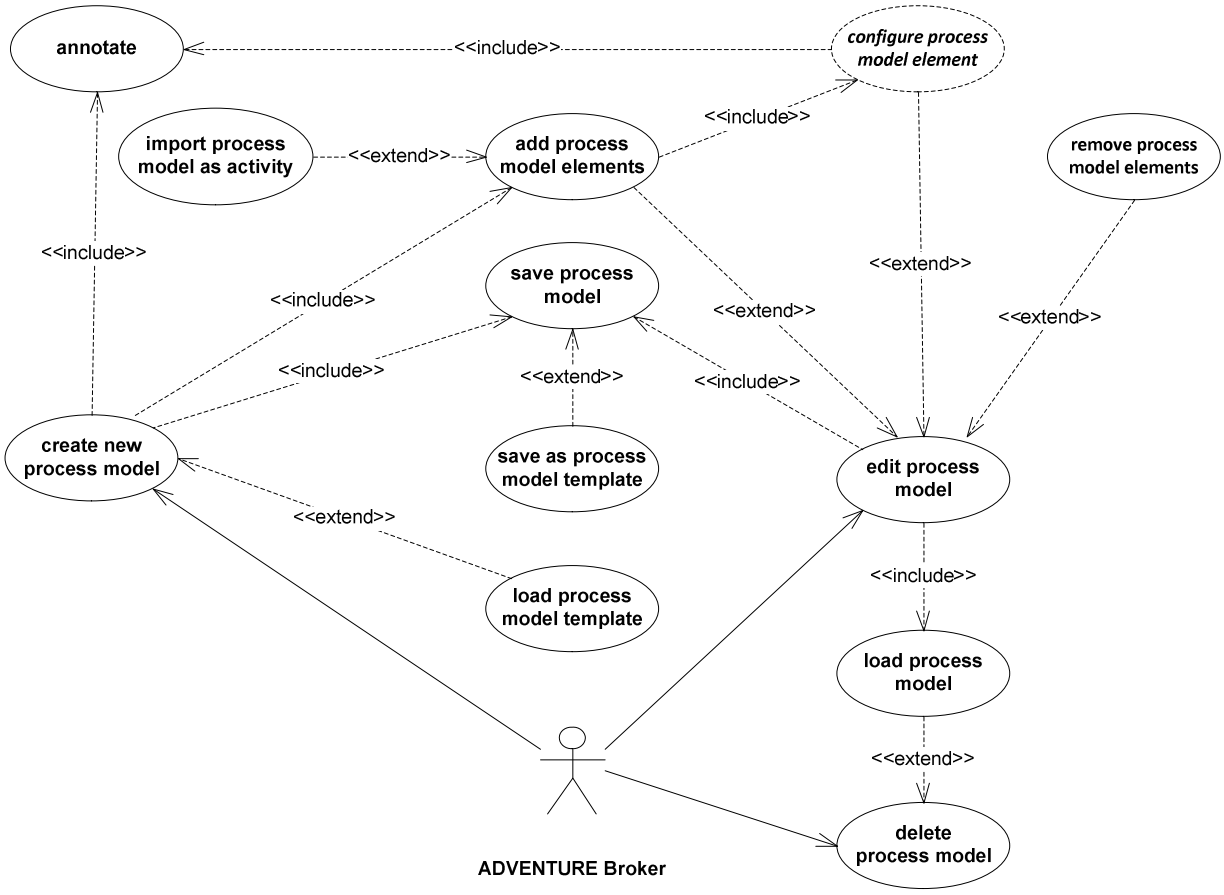

Figure 25 - Process Management Use Cases

Use Case: PD-U01	Name: Create New Process Model
Description	ADVENTURE broker wants to create a new process model from scratch or using an existing process template or process model.
Actors	ADVENTURE broker
Pre-conditions	The actor is authenticated and authorized for the operation (in role to create own process or process for another user)
Inputs	• ADVENTURE broker identity

Events sequence	• The actor invokes create new process model functionality on the Dashboard
	• Default: The actor chooses to create new process model from scratch
	  • The system displays an empty model for edit
	  • The actor fills in the process model high level metadata such as name, short description, etc.
	  • The actor adds process model elements: Includes PD-U08 “Add process model elements” Use Case.

	• Default: The actor saves the process model: Includes PD-U06 “Save Process Model” Use Case.
	• The actor may cancel the process at any time before saving.
	  • The actor is presented with the process editor in initial state.
	  • All changes are discarded
	• The actor chooses to save the process model as a template
	  • The system saves a new template: PD-U07 “Save as Process Model Template” Use Case
	• The actor chooses to create new process model from template
	  • The system loads the templates: PD-U02 “Load process template” Use Case

Extension points	• Use Case PD-U02 “Load process template”
Outputs	• New process model
Post-conditions	N/A
Requirements map	F3, F5, F47, F48, F51
Related Mock up	Figure 28 - New Process Model from Figure 29 - New Process Model from Scratch.

Use Case: PD-U02	**Name: Load Process Template**
Description	ADVENTURE Broker wants to create a new process model from an existing process template. ADVENTURE displays a list of available process templates.
Actors	ADVENTURE Broker
Pre-conditions	The actor is authenticated and authorized for the operation (in role to create own process or process for another user) Process templates are available for which the user has permissions to use
Inputs	Actor identity

Events sequence	• The actor invokes create new process model from template
	• Default: The system retrieves the process templates available for that actor.
	  • The actor chooses a process template to use from the list.
	  • The system displays a process model with its elements.
	• Process templates may not be available for this user.
	  • A notification message is displayed

Extension points	N/A
Outputs	Process template is displayed for further use
Post-conditions	N/A
Requirements map	F49, F50
Related Mock up	Figure 27 - Process Model Designer Entry

Use Case: PD-U03	Name: Edit Process Model
Description	ADVENTURE Broker wants to edit an existing process model.
Actors	ADVENTURE Broker
Pre-conditions	• The actor is authenticated and authorized for the operation (in role to create own process model or process model for another user)
Inputs	• Process model

Events sequence	• The actor invokes edit process model functionality on the Dashboard
	• The system checks whether the process model is in execution
	• Default: The system displays process model and its elements: Includes PD-U04 “Load Process Model” Use Case
	  • The actor edits the process model high level information such as name etc.
	  • The actor edits the process model.
	  • Default: The actor saves the process model: Includes PD-U06 “Save Process Model” Use Case.
	  • The actor may cancel the process at any time before saving.
	  • The actor is presented with the process editor in initial state.
	  • All changes are discarded
	• The process is in execution and cannot be edited
	  • The system notifies the actor that the process model cannot be edited

Extension points	• Use Case PD-U08 “Add process model element”. • Use Case PD-U10 “Configure Process Model Element”
Outputs	• Changed process model
Post-conditions	N/A
Requirements map	F3, F5, F7
Related Mockup	Figure 30 - Model Process Designer

Use Case: PD-U04	Name: Load Process Model
Description	ADVENTURE Broker wants to view an existing process model for editing or for creating a new process from it.
Actors	ADVENTURE Broker
Pre-conditions	• The actor is authenticated and authorized for the operation • Process models are available for which the user has permissions to use.
Inputs	• Actor identity
Events sequence	• Default: The actor invokes load process model

	functionality by selecting a process from a process list. • The system retrieves all the static information related to this process model, including process model elements and the services assigned accordingly. • The system checks the status of the process • The process is at design phase • The system displays a process model with its elements in the process editor. • The actor chooses to load (import) a process model from a known URI (file system) • The system opens a browse form • The user chooses a process model to load • The system loads the process model
Extension points	N/A
Outputs	• Process is displayed for further use
Post-conditions	N/A
Requirements map	F49
Related Mock up	Figure 27 - Process Model Designer Entry

Use Case: PD-U05	**Name: Delete Process Model**
Description	ADVENTURE broker wants to delete a process model
Actors	ADVENTURE broker
Pre-conditions	• The actor is authenticated and authorized for the operation (in role to delete own process model or process model for another user) • A process model to be deleted should be available and not in execution.
Inputs	• Existing Process Model
Events sequence	• The actor invokes delete process model functionality by selecting a process from a list in the Dashboard • The system checks whether the process is in execution • Default: The system is not in execution • The system requests confirmation from the actor • The actor confirms the deletion • Default: The system deletes the process model. • The actor may cancel the deletion • The actor is presented with the process editor in initial state. • The process is in execution and cannot be deleted • The system notifies the actor that the process cannot be deleted
Extension points	N/A
Outputs	• Process model deleted
Post-conditions	N/A
Requirements map	F49, F50
Related Mock up	Figure 27 - Process Model Designer Entry Figure 30 - Model Process Designer

Use Case: PD-U06	Name: Save Process Model
Description	ADVENTURE broker saves process model after creating a new one or editing an existing one
Actors	ADVENTURE broker
Pre-conditions	• The actor is authenticated and authorized for the operation • The actor should have filled the mandatory data correctly
Inputs	• Process Model

Events sequence	• The system makes a formal control on the data
	• Default: All the data is correct
	  • Default: The system saves the process model
	  • The system displays a confirmation message
	  • The data is not saved due to some unexpected problem
	  • The system displays an error message
	• There is an error in the data
	  • The system displays an error message

Extension points	• Use Case PD-U07 “Save as Process Model Template”
Outputs	• New or edited process model
Post-conditions	• The saved process model is available in the ADVENTURE system.
Requirements map	F49, F69
Related Mock up	Figure 30 - Model Process Designer

Use Case: PD-U07	Name: Save as Process Model Template
Description	ADVENTURE broker saves a process as a process template to be reused after creating a new one or editing an existing one
Actors	ADVENTURE broker
Pre-conditions	• The actor is authenticated and authorized for the operation • The actor should have filled the mandatory data correctly
Inputs	• Process Model

Events sequence	• The system makes a formal control on the data
	• Default: All the data is correct
	  • The actor defines the access rights for the process template. Default is Private
	  • Default: The system saves the process model as a template
	  • The system publishes (shares) the process template
	  • The system displays a confirmation message
	  • The data is not saved due to some unexpected problem
	  • The system displays an error message

	• There is some error in the data • The system displays an error message
Outputs	• Process Template
Post-conditions	• The saved process model template is available in the ADVENTURE system for use.
Requirements map	F50
Related Mock up	Figure 30 - Model Process Designer

Use Case: PD-U08	**Name: Add Process Model Elements**
Description	ADVENTURE Broker adds elements to a new or to an existing process model (this might be activity, connector, event, etc.)
Actors	ADVENTURE Broker
Pre-conditions	• The actor is authenticated and authorized for the operation • There is a process model created to which elements will be added
Inputs	• Element type

Events sequence	• The actor selects an abstract element to be added to the process model from a list of standard elements.
	  • The system opens the activity element configuration form: Includes “Configure Activity” Use Case.
	  • The system opens the event element configuration form: Includes “Configure Event” Use Case.
	  • The system opens the connector element configuration form: Includes “Configure Connector” Use Case.
	• The actor adds the element to the process model
	• Default: The actor configures the element: Includes PD-U10 “Configure Process Model Element” Use Case
	• The actor may cancel the process at any time before saving.
	  • The actor is presented with the process editor in initial state.
	  • All changes are discarded

Extension points	• Use Case PD-U09 “Import process model as activity”
Outputs	• Element described and added to process model
Post-conditions	N/A
Requirements map	F7, F47, F48, F53, F57, F62
Related Mock up	Figure 30 - Model Process Designer

Use Case: PD-U09	**Name: Import Process as Activity**
Description	ADVENTURE broker adds activity (sub-process) to a new or to an existing process model by importing an existing process model
Actors	ADVENTURE broker

Pre-conditions	• The actor is authenticated and authorized for the operation • There is a process model created to which activities (sub-processes) will be added
Inputs	• Partner process model

Events sequence	• The actor browses the existing and available process models
	• Default: The system displays a list of process models.
	  • The actor selects a process model to be added as activity (sub-process) in his/her process model
	  • The system retrieves the chosen process model
	  • Default: The actor adds the element (sub-process) to the process model
	  • The actor may cancel the process at any time before saving.
	  • The actor is presented with the process editor in initial state.
	  • All changes are discarded
	• The system does not find available process models for this actor
	  • The system displays an empty list.

Extension points	• Use Case PD-U10 “Configure Process Model Element”
Outputs	• External process model added as activity to process model
Post-conditions	N/A
Requirements map	F49
Related Mock up	Activity based on an existing partner process model.

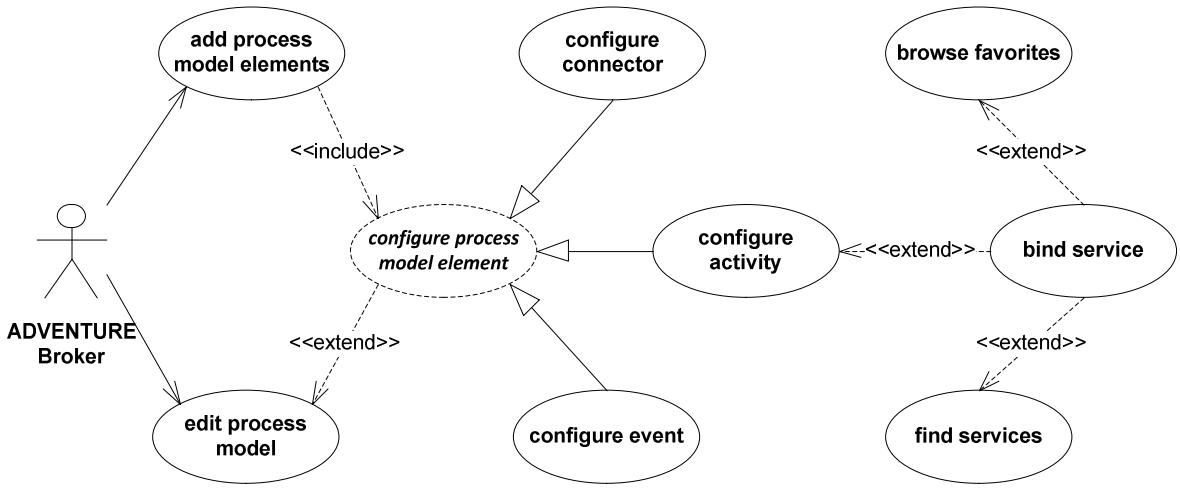

Figure 26 - Configure Process Model Elements Use Cases

Use Case: PD-U10	Name: Configure Process Model Element
Description	ADVENTURE broker configures a new or changed process element (this might be activity, event or connector)
Actors	ADVENTURE broker
Pre-conditions	• The broker is authenticated and authorized for the operation • There are elements created that will be configured.
Inputs	• Element from the process model

Events sequence	• The broker invokes the element configuration functionality
	• The system opens the element configuration form:
	  • The system opens the activity element configuration form:
	  • The system opens the event element configuration form: Includes “Configure Event” Use Case.
	  • The system opens the connector element configuration form: Includes “Configure Connector” Use Case.
	• The broker fills in the element information

Extension points	• Use Case PD-U11 “Bind service”
Outputs	• Configured process model element
Post-conditions	N/A
Requirements map	F7, F47, F48, F53, F57, F62, 79
Related Mock up	Figure 30 - Model Process Designer

Use Case: PD-U11	Name: Bind Service
Description	ADVENTURE broker binds a service to a new or to an existing process activity
Actors	ADVENTURE broker
Pre-conditions	• The actor is authenticated and authorized for the operation • There is a process and an activity created to which services will be bound
Inputs	• Process activity

Events sequence	• Default: The broker invokes “bind service” functionality
	  • The system searches for suitable services based on the criteria defined during process activity configuration:
	  • The system retrieves all suitable matches
	  • The system displays a list of matching services
	  • The broker chooses a service to bind
	  • Default: The system binds the service to the respective element.
	  • The broker may cancel the process at any time before saving.
	  • The broker is presented with the process editor in

	initial state • All changes are discarded • The broker browses a list of favourites (this might include partners profiles or services) to selects an appropriate service to be bind to the element: Includes PD-U12 “Browse Favourites” Use Case
Extension points	• Use Case PD-U12 “Browse Favourites”,
Outputs	• Service bind to process element
Post-conditions	N/A
Requirements map	F13, F31, F47, F48, 52
Related Mock up	Figure 31 - Partner Assignment to

Use Case: PD-U12	**Name: Browse Favorites**
Description	ADVENTURE broker binds service to process activity by selecting a service from an existing list of favourite services / partners
Actors	ADVENTURE broker
Pre-conditions	• The actor is authenticated and authorized for the operation • There is a process and an activity created to which services will be bound.
Inputs	• Process element, Actor identification
Events sequence	• The broker invokes “browse favourites” functionality. • The system searches for the services/partner profiles marked as favourite for this actor. • Default: The system retrieves all favourite services/partner profiles for this broker. • The system displays a list of all favourite services/partner profiles for this broker. • The system does not find “favourites” for this actor • The system displays an empty list
Extension points	N/A
Outputs	• List of all favourite services/partner profiles for this broker
Post-conditions	N/A
Requirements map	F12, F13, F56
Related Mock up	N/A

5.2.2.1 User Interface Mockups

Following, example mock ups are presented in order to visualise better the process designer user interface and it functionality.

5.2.2.1.1 Process Model Designer entry page

Once the user accesses the Process Model Designer functionality from the Dashboard, it can access a list of the different process models it has designed and saved into the system.

From this initial screen, the user will be able to find a specific process model and proceed to work on it. There can be different types of process models accessible from this screen:

- Process Models generated and saved from the Process Optimization functionality.
- Process Models that have been saved from the Process Forecasting and Simulation functionality.
- Process Models that have been designed by the user and have been saved for further usage (or that are deployed into the Adaptive Process Execution environment).

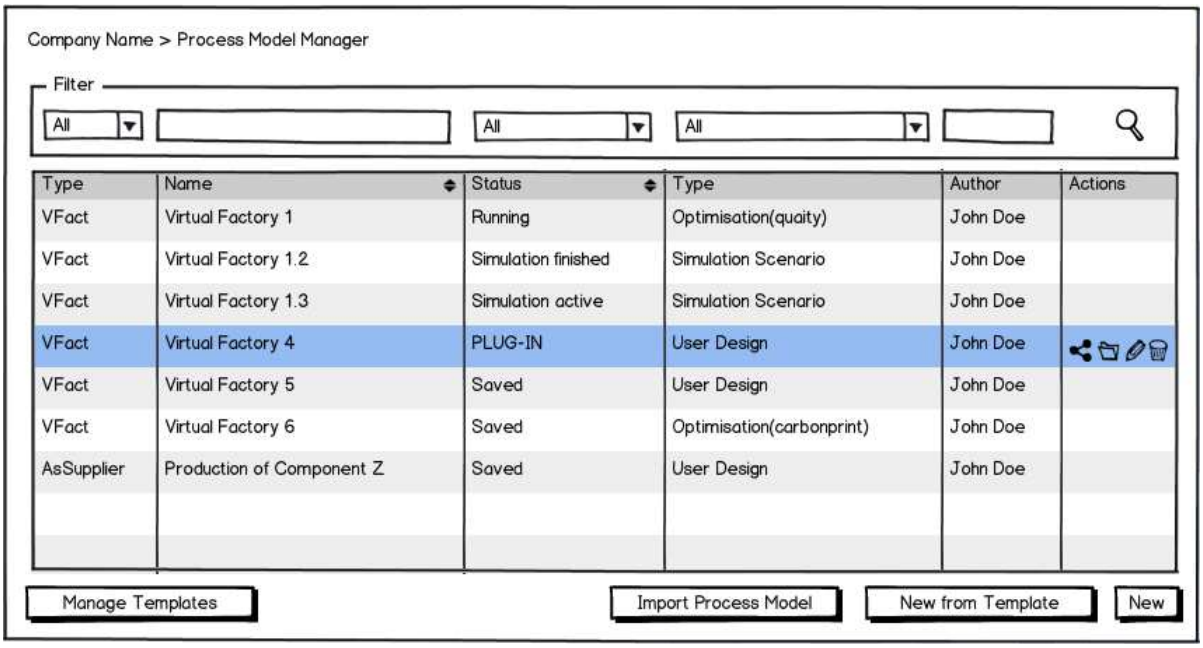

Figure 27 - Process Model Designer Entry Page

From the Process Model Designer entry screen, the users can open a process model diagram, can edit or can just delete it (depending on the execution status, as stated in the previous section).

The user also can share a specific process model so that it can be used by third parties as an activity into their process models (e.g. a virtual factory). When the user shares a process model, it can give permission to specific companies, or can make it public to be used by

anyone at ADVENTURE. This functionality relies on Data Provision and Discovery component.

It is important to highlight that the ADVENTURE project intention is not to develop just another enterprise process editor, but an environment in which a company should be able to reuse its existing process models and share them with other companies in a dynamic way. This will allow (small) companies to be able to coordinate and monitor complex collaboration scenarios (e.g. manufacturing) by means of composing existing specialized companies' services. The Process Editor (and its auxiliary tools) will be the mechanism that will allow a company to find the appropriate companies (and services) in order to optimize, expand or create its new extra-company collaboration scenarios (Virtual Factories).

Taking this into account, the process editor will allow companies to build their specific process models (or import them) with basic process editing capabilities (so that it will be easy for SMEs to use them for composing their advanced collaboration scenarios).

5.2.2.1.2 New Process Model

The UI mock up here applies to **PD-U01** (Create New Process Model) use case.

Starting from the previous screen, the user can start a NEW process model design. There are two possibilities for creating a new design:

- Start from a saved template.
- Start from scratch.

In the first case, the user can reuse process model templates proposed by the system, saved by itself (or a user of its company), or templates that have been shared by third companies.

A process template is a process model with its modelling elements (activities, events, flows, etc.), but with no or incomplete assignments to partners that have to execute the different activities. Technically, any process can also be a template.

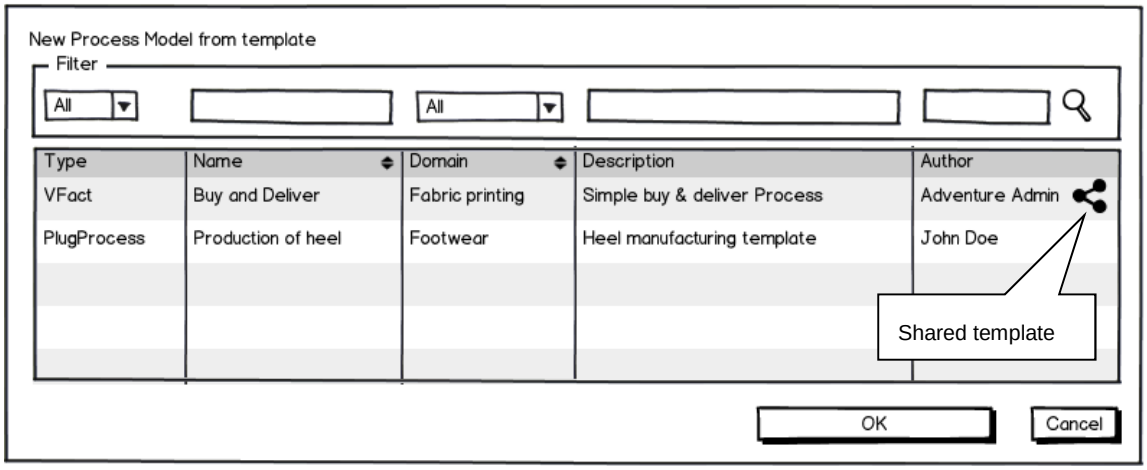

Figure 28 - New Process Model from Template

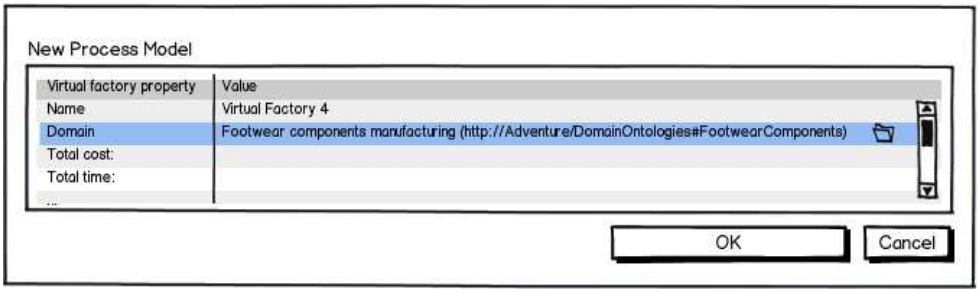

Figure 29 - New Process Model from Scratch.

In both cases, some high level information about the new model will be provided by the user, including the characterization on the process model (semantic annotation) based on existing ontologies in the ADVENTURE domain. After this information is provided, the Process Model Designer will be opened.

If the user is creating a new process model from scratch, an empty model will be shown. In case the user is starting from a template, the activities saved in the template will be loaded in the editor.

5.2.2.1.3 Manage Templates

The templates management functionality will allow the users to perform different actions on the saved and shared templates.

Users will be able to edit or delete saved templates. This action will not affect the process models based on these templates as they use the template as a base for designing the process model, being from this moment, completely different copies of information. The template manager will allow a user to rate and share templates (or stop sharing them).

5.2.2.1.4 Import Process model

In the market there are many existing good tools that are able to deal with the process design activities. The intention of the ADVENTURE project is not to force the companies to use the ADVENTURE Process Editor but on the other hand ADVENTURE adoption implies any process must be in a format understandable by the ADVENTURE components. So if a company already has its processes modelled, or they just prefer an offline editor they can use them to model the basic process design features, import it (if is in an ADVENTURE compatible format), and then finalise the activities configuration by using the ADVENTURE Process Editor.

The ADVENTURE platform will specify at a further stage (in the technical specification), which are the data formats that the platform will be able to import (and in particular which data subsets).

Once the process is (optionally) designed in an alternative editor, it has to be imported into the ADVENTURE process editor in order to configure the different activities in terms of which partner has to execute a specific task, what is the semantic characterization (annotation) of an activity X, etc.

The ADVENTURE platform will provide from this moment on, the capabilities of simulation, optimization, plug, etc. of the imported process model.

Once the process model is imported, the user can also use the Process Model Designer to perform editing actions on it, but after finishing all the editing and configuration stage at the ADVENTURE platform, the resulting model will not be exportable to the original (offline) editor (since it will contain specific ADVENTURE information that the external editor will not able to cope with).

Some high level information about the new model will be provided by the user at the import stage, including the characterization on the process model based on existing ontologies in the ADVENTURE domains. After this information is provided, the Process Model Designer will be opened.

5.2.2.1.5 Process Model Designer Screen

The Process Model Designer screen allows the users to design and configure their process models in a graphical way. The UI mock-ups here apply to the **PD-U03** (Edit Process Model) use case.

Apart from this main functionality, this screen gives the user access to the following functionalities:

- Launch (and/or save) the edited process model into the Process Forecasting and Simulation, and Process Optimization functionalities.
- Access (through the top tabs) to the results of the simulations and optimizations performed on the designed process model.
- Deploy the edited process into the execution environment (so triggering the PLUG processes at the Process Monitoring functionality).

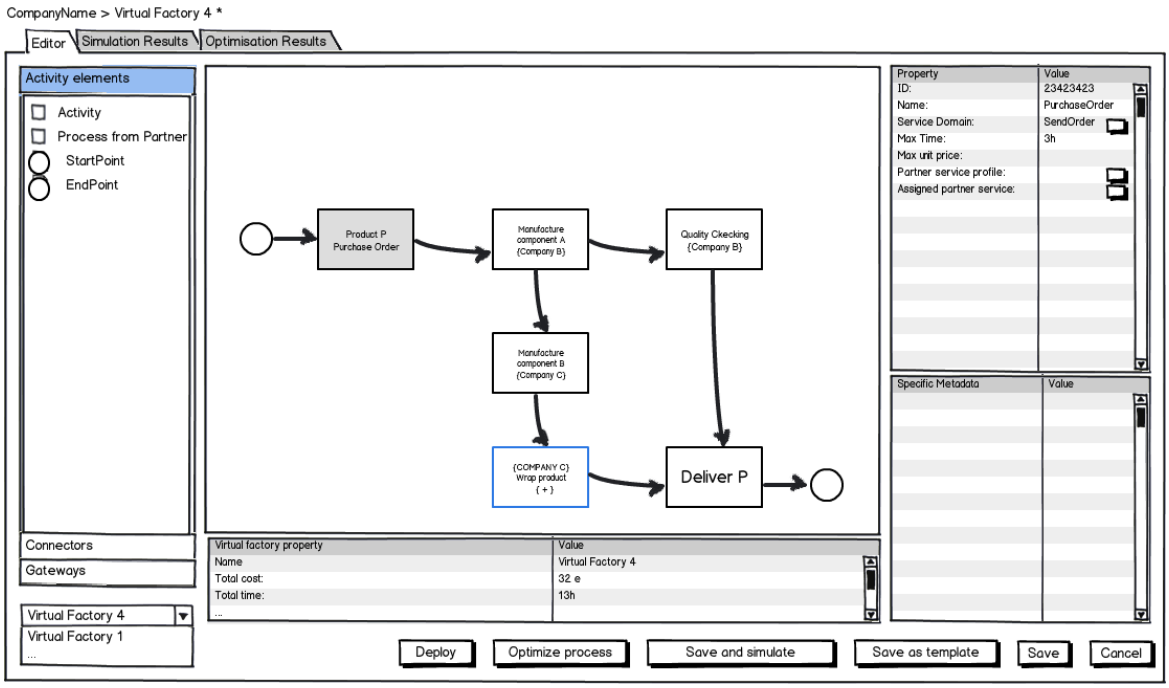

Figure 30 - Model Process Designer

The editor screen is divided into different functional areas:

- The Activity components collections: (top left) contains all the possible graphical elements that the user can select to design a process model in the graphical design canvas.
- The graphical design canvas: (centre) allows the user to drop elements from the activity components collections in order to draw the whole process model.
- The overall properties editor: (bottom centre) permits the user to provide the overall properties for the edited process model.
- Specific activity properties editor: Located on the right side of the screen, allows the user to configure individual Activities with the operational data needed (max time to execute, price, partner profile, etc...). This part will interact with the user interface of the data provisioning and discovery functionality, allowing the user to search for partners, process models, ontology description of the processes, partners generic profiles, etc... This part is split into the following:
    - Providing generic activity information
    - Allowing the user to define and manage specific metadata for each activity, so having the opportunity to semantically annotate the activity
- User Process Navigator: Located on the bottom left side, can be used to quickly change from a Process Model to a different one.
- Action Buttons: Located at the bottom part, can be used to perform the following action on the edited process:
    - Deploy: Interact with the Monitoring and Execution functionality in order to deploy the currently edited process (though the PLUG processes)
    - Optimize process: Save and launch the currently edited process into the process optimization engine
    - Simulate: Save the process and launch it (with its current configuration) into the Process simulation engine
    - Save as Template: Save the designed process model, so that it can be used as a template in the future. When a user saves a process model as a template, it will have the opportunity of sharing this template with other companies or with the whole ADVENTURE community
        - Save: Keep the process model for further usage
        - Cancel: Discard the changes made on the model since the last save action

On top of the screen, there are different tabs, allowing the user access to the results of the simulation and optimization processes made on the currently edited model.

5.2.2.1.6 Partner Services Assignment to Activities

In order to configure the Process Model to be executed or simulated, the different activities have to be bound to the partners (or partner service profiles) intended to perform them.

Specifically, the user assigns the partner services or partner services profiles intended to be used in the execution of specific activities at the process model.

This action involves accessing the Data Provisioning and Discovery component which will provide the search features and information needed for the user to find the appropriate partner service or partner service profile to cope with the activity designed. In fact many of the functionalities that can be assigned to a specific Activity will be found by means of the Data Provisioning and Discovery features. The Data Provisioning and Discovery module have access to the different services and configurations available at the ADVENTURE platform, so the Process Model Editor will make use of it in order to find specific data gateways, core messaging functionalities, etc. that can be used at the activities configuration.

The partner service search dialog will use the already configured features of the target activity in order to constrain the list of candidates able to perform it (type of process, Max time, etc.). The search dialog will be shown when a user clicks on the buttons presented at the activity configuration (see orange box and small orange button in the following figure).

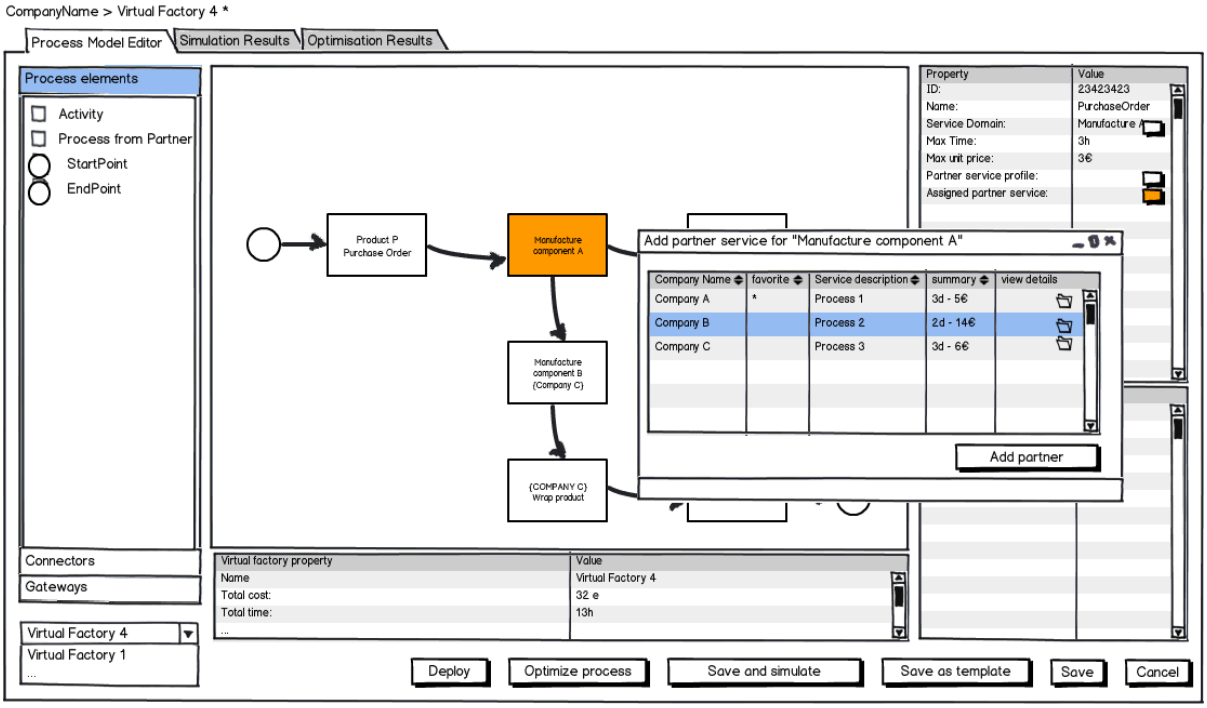

Figure 31 - Partner Assignment to Activity

Once the user chooses the appropriate partner to deal with the activity, they assigns it to the activity by pressing the Add Partner button.

Before assigning a partner, the user can have access to further information about the companies by selecting the view details button associated to each company.

Company services can be rated and configured as favourites, in this way when search functionalities are shown, the user have information about which ones are its favourites, and order the list of partner services by this criteria.

5.2.2.1.7 Partner Service Profile Assignment to Activities

The user will also be capable of assigning a generic partner service profile instead of a specific partner to deal with specific activities (e.g. select "Chair manufacturing" instead of a specific company service, from an existing ontology at ADVENTURE). This functionality will afterwards (as an example) allow the simulation functionality to simulate the activity over the different candidates that offer the same kind of service (e.g. the different Chair manufacturers present at the system). In this way the user will be capable of making the partner selection based on the simulation results.

The Process Model Designer will interact with the Data Provisioning and Discovery functionality in order to obtain the different partner profile list, services, and description.

5.2.2.1.8 New Activity based on an existing Process Model from a partner

Instead of creating a new generic activity, the user can also make a search for partners (before creating an activity), access the process models description of the specific partners, and if one of them fits with the activity to be designed then selects it. For this purpose the process model descriptions should be public or at least shared for this user.

In this way, a complete Process Model from the partner will be "imported" as an Activity into the user new Process Model. (e.g. for a t-shirt manufacturer, the "fabric printing" is just an activity that can be based on the whole process model of a company that offers "fabric printing").

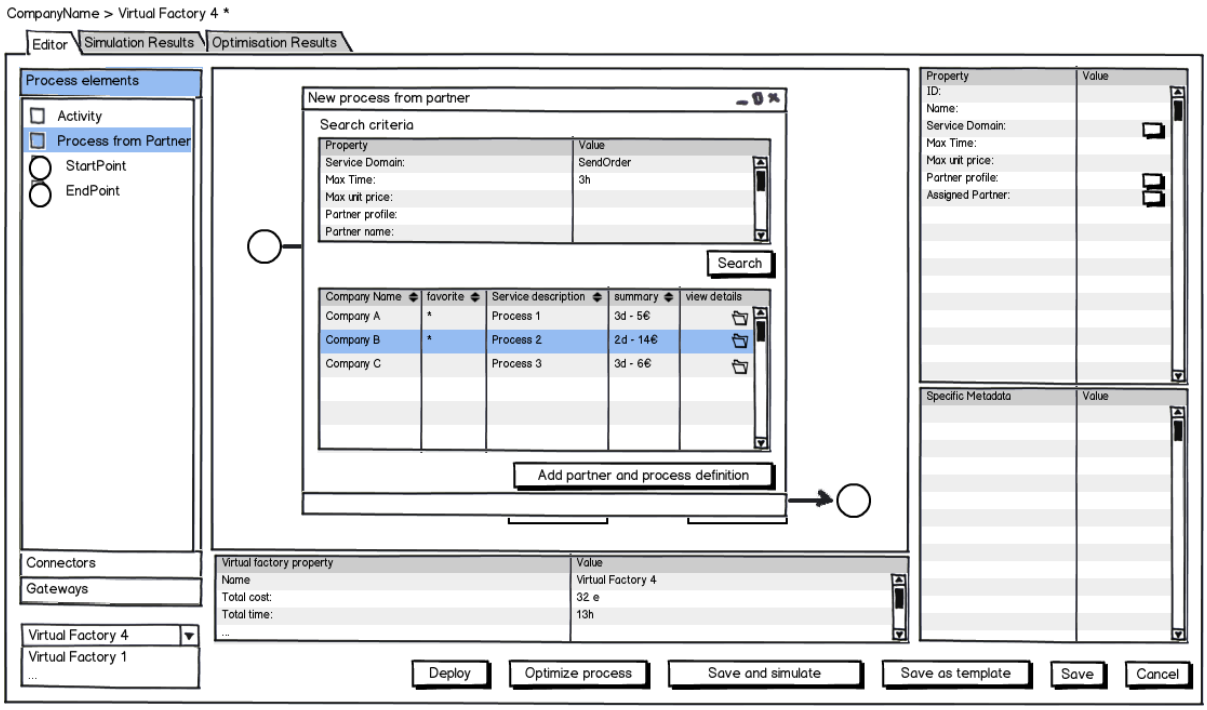

Figure 32 - Activity based on an existing partner process model.

The Process Model Designer will interact with the Data Provisioning and Discovery functionality in order to obtain the partners lists, processes managed by the partners and the descriptions.

The activities based on partners process models, will be represented in a different way at the graphical design canvas.

5.2.3 Component Interaction

The Process Designer component realizes the ADVENTURE use cases for interactive Smart Process definition, publication and maintenance. In order to provide this functionality the Process Designer component interacts with other ADVENTURE functional components as follows:

- *Dashboard* component provides the Process Designer UI *with a holistic interface with the rest of ADVENTURE components,* allowing the user to effectively define, publish and maintain the ADVENTURE Smart Processes
- *Data Provisioning and Discovery* component provides the data for defining smart processes including process models, process activities, services to bind to these process activities, process rules and constraints, as well as the corresponding annotations.
- *Process Forecasting and Simulation* component makes simulative performances of the process model, being defined by the Process Designer, allowing the user to obtain preliminary results for the process model for certain conditions.
- *Process Execution* component executes instances of the process model, defined by the Process Designer.
- *Adaptation* component adapts the process defined by the Process Designer to new conditions.
- *Process Optimisation* component allows the user to construct via the Process Designer the optimal manufacturing process according to different criteria.
- *Process Monitoring* component allows the user to observe the process performance and to make changes when necessary.

The diagram below summarizes the dependences between Process Designer and other ADVENTURE functional components.

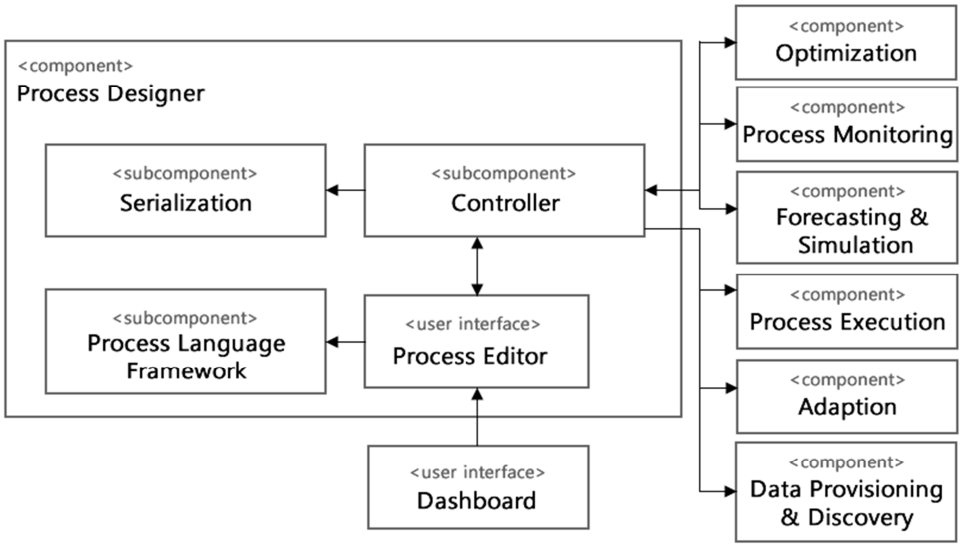

Figure 33 - Process Designer Component Interaction

5.2.4 Conceptual Data Model

The key domain model in Process Designer's scope is the Process Model. The diagram on Figure 34 depicts the types of relationships it has in this scope.

Process model:

- A *Process Model* contains one or more *Modelling Elements* (according to the chosen format and notation, which will be selected in the technical specification) and may be provided as a template
- A specific Modelling Element called Activity can be bound to zero or more service operations (provided by one or more ADVENTURE members)
- A Process Model may be shared with zero or more ADVENTURE members (each member could have zero or more models shared for it)
- Zero or more ADVENTURE members may be partners in a collaborative process (depending on whether their services are chosen to be bound to activities in a model)
- At least one ADVENTURE member is in the role of a process broker.
- At least one (and possibly more) ADVENTURE members can own a process model (manage its lifecycle) and each can own multiple processes.

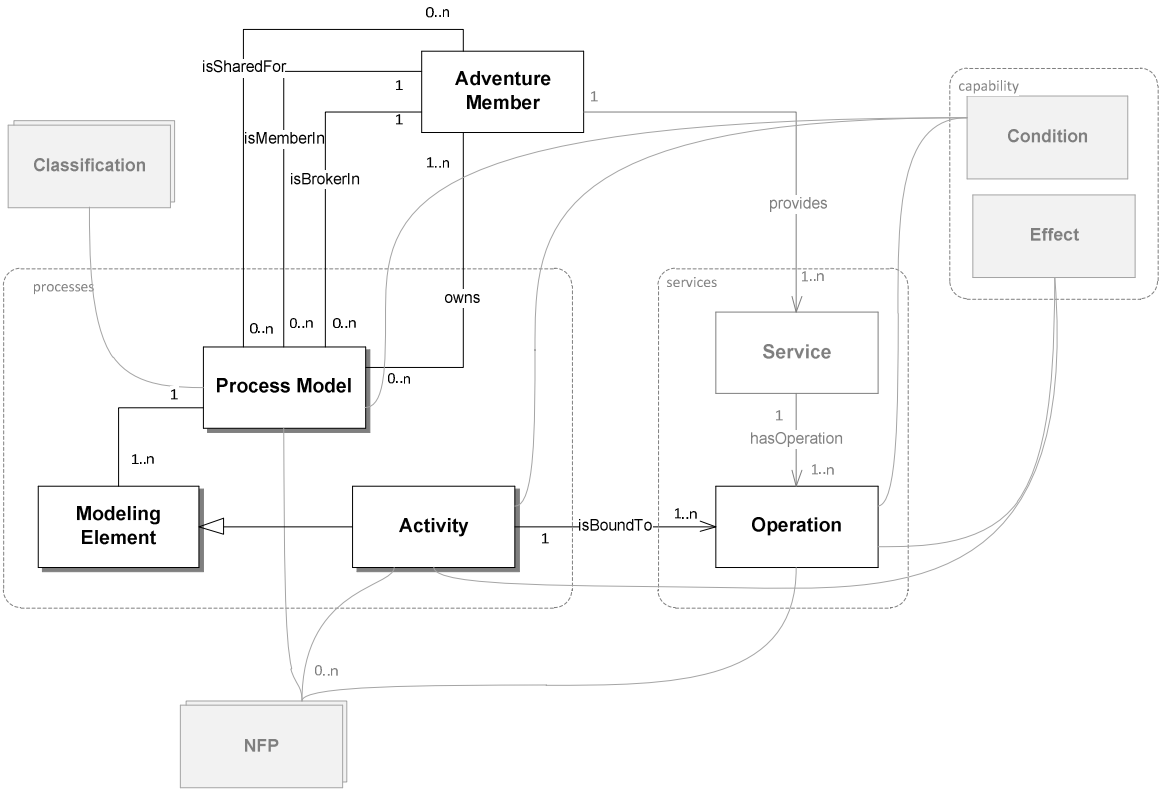

Figure 34 - Process Designer Conceptual Data Model

***Note**: NFP is an acronym of Non Functional Properties*

5.2.5 Parameters to Take into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	++
Regularly Updated	+
Technical Up-to-Datedness / Appeal	+
Open Source	+++
Non-Infecting	+++
Code-Quality	++
Extensibility	++
Community	+++
Performance	+/-
Reuse of existing developments	+/-
EU project origin	+/-
Platform (Portability)	+
Open Standards Compliance	++
Interoperability	++
Specific Parameters	
Generation of executable models	+++
Web based editor	+++
Parameter mapping	+
Support for Multi-Binding	++
BPMN elements support	++
Decision criteria definition	+
Calling parameter definition	+
Usability	+++
Model validator	++
Script/code editor	+
Basic events support.	++
Support rules definition	+/-
Human task support	+/-
Strictly block structured process logic	+/-
Export to image/vectorial formats (SVG, png, ...)	+

5.3 Cloud Storage

The Cloud Storage component is located in the Data Management layer of the ADVENTURE architecture and serves as central data storage. It allows each component to store different types of data (binary, semi-structured, semantic) and will ensure high scalability for data storage processes. Within this component, scalability and reliability both have a higher priority than speed.

5.3.1 Overall Functional Characterization

The Cloud Storage component will be based on buckets, which are specific isolated storage spaces managing data for one or more components (see Figure 36 - Architecture of the Cloud Storage Component). It has a “bucket type”, which defines its core features (e.g. a binary bucket for documents or a semantic bucket for semantic data) and which will allow components to select the optimal storage type for their needs. Each component may create multiple buckets for data storage, which are fully separated and not influenced by activities of other buckets. For example, if a component wants to store binary and semantic data it creates a binary and a semantic bucket. These buckets can be thought as independent databases. To store and retrieve binary data the example component has to address the binary bucket and to store and retrieve the semantic data it has to address the semantic bucket. Thereby the component can decide oneself, how it stores the data.

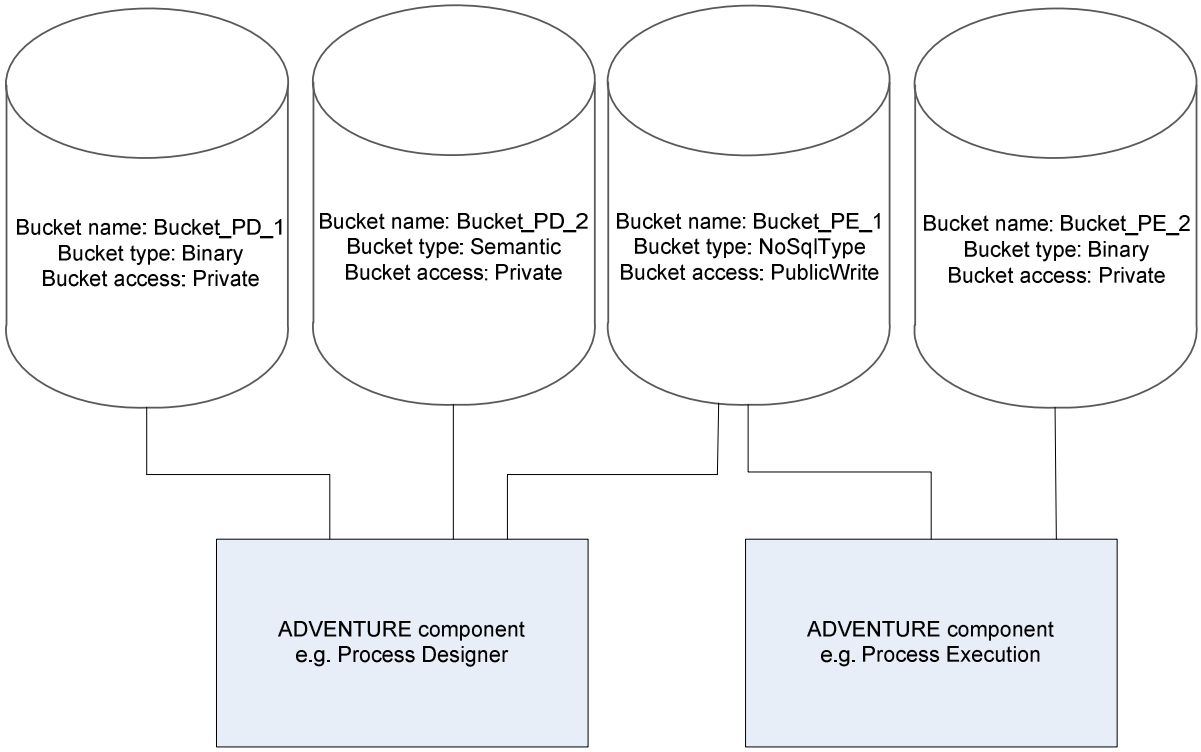

Figure 35 - Isolated Buckets

Within the project, a set of three different bucket types for semi-structured, semantic and binary data will be implemented, but this approach is very flexible, so that further bucket types, as for example a SQL bucket, can be added easily The Cloud Storage will support a

basic set of CRUD (create, read, update, delete) operations for all bucket types in a suitable data format (e.g. OData for structured data).

Additionally the Cloud Storage component will provide a simplistic access control list-feature (ACL), which may optionally be used by components.

The security of the Cloud Storage component will partially be based on the ACL list, including different roles represented by components, e.g. allowing the optimization component to access the process designer buckets with read and write access while allowing all other components read-only access. The general access can be set by an access level like for example PublicRead or PublicWrite. This procedure allows using buckets across several components. For example Process Designer and Process Execution can share a bucket to store the Process Model.

Additionally the Cloud Storage will use the ACL to provide user based authorized access to binary data (e.g. documents or images). If necessary the authentication for other data has to be implemented in the components that use the data, because it is technically not possible to perform an authentication check on structured or semantic data because of relations.

In case of data storage errors, as corrupted data or faulty database communication, the Cloud Storage component will try to automatically handle them if possible. For those cases, where an error cannot be handled automatically the component will return fault messages to the component that has launched the request.

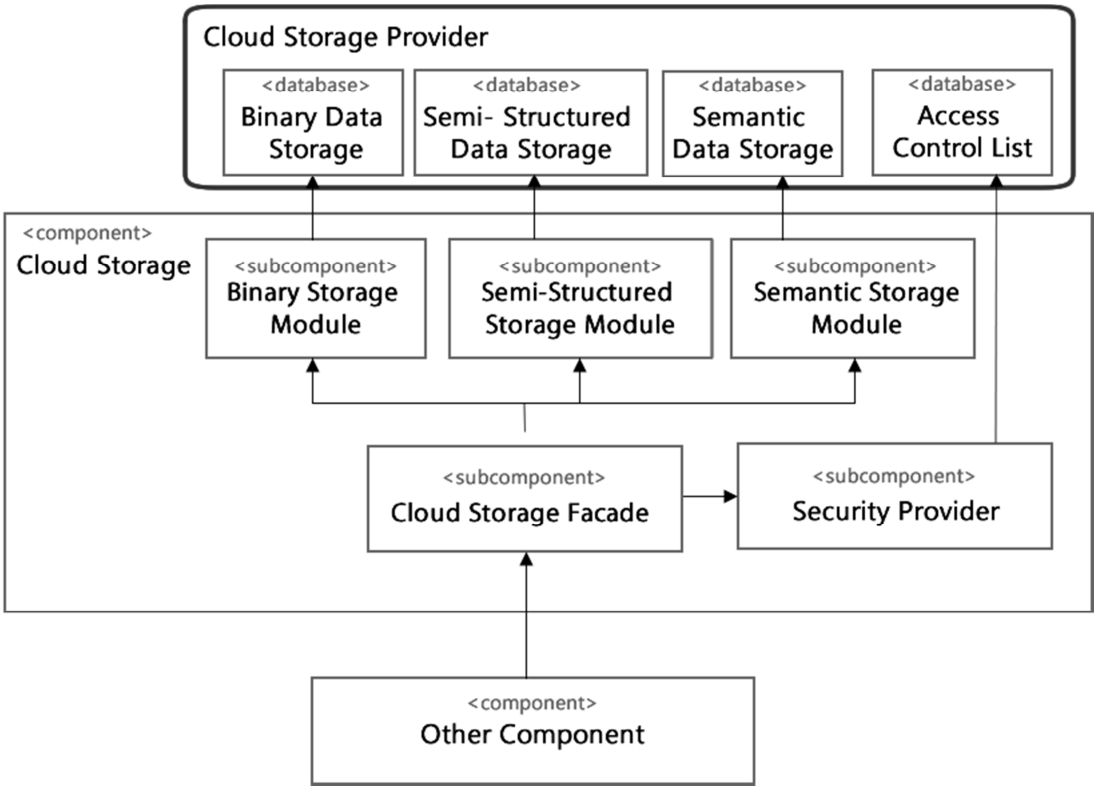

Figure 36 - Architecture of the Cloud Storage Component

The Cloud Storage component consists of several sub components:

- **Cloud Storage Facade** – This is the message interface of the Cloud Storage component. It receives the queries, checks access rights and sends the results back. For authentication checks and ACL queries it invokes the Security Provider subcomponent.
- **Binary Storage Module** – This module is an interface to the Binary Data Storage. It executes the binary queries and returns the results to the Cloud Storage Facade.
- **Semi-Structured Storage Module** – This module is an interface to the Semi-Structured Data Storage. It executes the semi-structured (e.g. for NoSQL databases) queries and returns the results to the Cloud Storage Facade.
- **Semantic Storage Module** – This module is an interface to the Semantic Storage. It executes the semantic queries and returns the results to the Cloud Storage Facade.
- **Security Provider** – It provides a simplistic authentication mechanism that is based on an access control list-feature (ACL) and can be queried over the in the Cloud Storage Facade.

5.3.2 Use Cases

The following functionalities have been identified from the requirements:

- Store semi-structured, semantic and binary data

  The Cloud Storage will essentially provide CRUD functionality: create, read, update and delete. It will be possible to store different predefined and not predefined semi-structured data like company profiles, service descriptions, relationships, monitoring data, processes, etc., semantic data like service descriptions and binary data like documents or images.

- Discover and provide specific semi-structured, semantic and binary data

  The Cloud Storage will provide a possibility to query the semi-structured, semantic and binary data and returns the results. It will provide basic CRUD operations for these data types.

- Authentication system for access

  The Cloud Storage will manage a simplistic access control list-feature (ACL), which can be used by components and will be used by the Cloud Storage component as the first part of the internal bucket and binary data security. The second part of bucket security will be realised with a general access level like for example PublicRead or PublicWrite.

- Version control for binary storage

  The binary storage will provide a version control to restore older versions of documents or other binary data.

- Data archiving engine

  The data archiving engine contains an archiving policy if and how long data will be stored and applies the changes. Because this requirement has a lower priority, it is possible that this feature will not be implemented.

  - **Links between different storage types**

The possibility to link between the different data types, such as documents to semi-structured data have to be implemented in the components that need this feature. The Cloud Storage can store the links, but they have to be created in the stored data by the components that use this feature and maintained.

A user interfaces are implied by the functionalities

- **Administration user interface**
  - An administration user interface will be needed to configure the Cloud Storage and which will be embedded in the Dashboard.

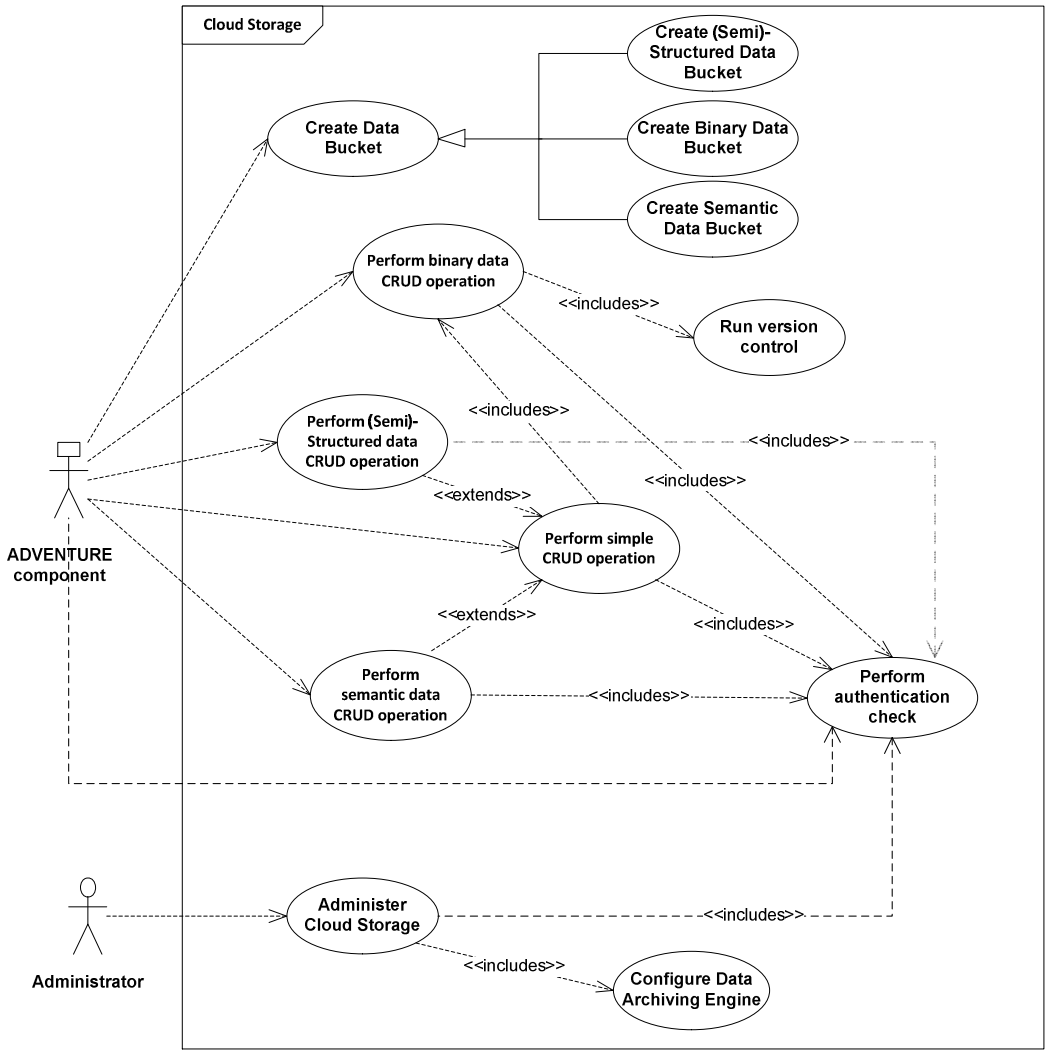

Figure 37 - Cloud Storage Use Case Diagram

Use Case: CS-U01	Name: Create Data Bucket
Description	If an ADVENTURE component wants to store something in the Cloud Storage, it needs a Data Bucket, where it has write permission. Each component can create multiple Data Buckets of different types to store semi-structured, semantic or binary data.
Actors	ADVENTURE component
Pre-conditions	N/A
Inputs	• Bucket Type • Component Identifier
Events sequence	• A component sends a message with the bucket type, component identifier and the command to create a bucket to the Cloud Storage. • The Cloud Storage component creates the bucket. • The Cloud Storage sets the corresponding value in the ACL that the component can access the bucket. • The Cloud Storage sends the message with the bucket identifier back to the component that sends the message to create the bucket.
Extension points	N/A
Outputs	• Bucket Identifier
Post-conditions	N/A
Requirements map	N/A
Related Mock up	N/A

Use Case: CS-U02	Name: Perform simple CRUD operations
Description	The Cloud Storage will provide simple CRUD (create, read, update, delete) operation for all types of Buckets, such as selecting or deleting object with a certain Id.
Actors	ADVENTURE component
Pre-conditions	• The bucket has to exist
Inputs	• Bucket Identifier • CRUD command • Object Identifier
Events sequence	• A component sends a message to create, read, update or delete an object with a specific identifier. • The Cloud Storage performs an authentication check, if the component can access the bucket. • The Cloud Storage sends an error message, if the component has not the needed rights • Default: The Cloud Storage performs the query. • The Cloud Storage sends the result back.
Extension Points	• Use Case: CS-U03: Perform (Semi-) Structured data CRUD operations • Use Case: CS-U04: Perform binary data CRUD operations • Use Case: CS-U05: Perform semantic data CRUD

	operations
Outputs	• For create, update or delete a Boolean, if the operation was successful. • For read the object with given identifier.
Post-conditions	N/A
Requirements map	N/A
Related Mock up	N/A

Use Case: CS-U03	**Name: Perform (Semi-) Structured data CRUD operations**
Description	Additional to the basic set of CRUD operation from the simple CRUD operations use case, which will be essentially provided by the Cloud Storage, it will be possible to create more complex CRUD operations in a suitable data format as for example OData to make an accurate selection with where clauses possible.
Actors	ADVENTURE component
Pre-conditions	• The bucket has to exist
Inputs	• Bucket Identifier • Specific Query
Events sequence	• A component sends a query message. • The Cloud Storage performs an authentication check, if the component can access the bucket. • The Cloud Storage sends an error message, if the component has not the needed rights • Default: The Cloud Storage performs the query. • The Cloud Storage sends the result back.
Extension Points	N/A
Outputs	• Result
Post-conditions	N/A
Requirements map	F1, F2, F3, F5, F6, F8, F9, F10, F11, F12, F13, F21, F22, F23, F24, F26, F31, F34, F35, F36, F43, F44, F46, F47, F49, F50, F52, F56, F57, F59, F62, F73, F76, F77, F78, F80, F83
Related Mock up	N/A

Use Case: CS-U04	**Name: Perform binary data CRUD operations**
Description	The simple CRUD operations from use case CS-U02 will cover all CRUD operations that can be executed on binary data. For binary data the Cloud Storage will perform an user based authentication check.
Actors	ADVENTURE component
Pre-conditions	• The bucket has to exist
Inputs	• Bucket Identifier • CRUD command

	• Object Identifier • User Identifier

Events sequence	• A component sends a message to create, read, update or delete an object with a specific identifier.
	• The Cloud Storage performs an authentication check, if the component can access the bucket.
	  • The Cloud Storage sends an error message, if the component has not the needed rights.
	• Default: The Cloud Storage performs an authentication check, if the user can access the binary file.
	  • The Cloud Storage sends an error message, if the user has not the needed rights.
	• Default: The Cloud Storage performs the query.
	• The Cloud Storage sends the result back.

Outputs	• For create, update or delete a Boolean, if the operation was successful. • For read the object with given identifier.
Post-conditions	N/A
Requirements map	F53, F68
Related Mock up	N/A

Use Case: CS-U05	**Name: Perform semantic data CRUD operations**
Description	Additional to the basic set of CRUD operation from the simple CRUD operations use case, which will be essentially provided by the Cloud Storage, it will be possible to create more complex semantic CRUD operations in a suitable data format as for example SparQL to perform complex semantic queries.
Actors	ADVENTURE component
Pre-conditions	• The bucket has to exist
Inputs	• Bucket Identifier • Specific Semantic Query
Events sequence	• A component sends a semantic query message. • The Cloud Storage performs an authentication check, if the component can access the bucket. • The Cloud Storage sends an error message, if the component has not the needed rights • Default: The Cloud Storage performs the query. The Cloud Storage sends the result back.
Extension Points	N/A
Outputs	Result
Post-conditions	N/A
Requirements map	F70, F71, F72
Related Mock up	N/A

Use Case: CS-U06	Name: Run version control
Description	For binary data there will be a version control, which manages the versions of the documents.
Actors	Cloud Storage
Pre-conditions	• Version Control has to be enabled • Binary file has to exist in the bucket
Inputs	• Binary Data • Object Identifier • Bucket Identifier
Events sequence	• A component sends a binary file to update it. • Cloud Storage creates a backup file of the last file version. • Cloud Storage stores the new binary file.
Extension Points	N/A
Outputs	N/A
Post-conditions	N/A
Requirements map	F67
Related Mock up	N/A

Use Case: CS-U07	Name: Perform authentication check
Description	The Cloud Storage will manage a simplistic access control list (ACL), which can be queried by other components. The Cloud Storage uses this list to check for each access the authentication on a bucket level and for binary data. Additionally to the access list the Cloud Storage will use general access levels for buckets.
Actors	ADVENTURE component, Cloud Storage
Pre-conditions	None
Inputs	User or Component Identifier
Events sequence	• A component sends an ACL request message. • The Security Provider subcomponent (SP) receives the message. • SP checks, if the user or component identifier exists. • If the identifier not exists, SP returns an error message. • Default: SP reads the roles and granted rights from the storage. • SP returns the result.
Extension Points	N/A
Outputs	List with roles and granted rights
Post-conditions	N/A
Requirements map	F54
Related Mock up	N/A

Use Case: CS-U08	Name: Administer cloud storage
Description	An administration user interface will be provided, which

	allows administrators to customise the Cloud Storage and the authentication subcomponent. Additionally it provides the possibility to manage the Access Control List.
Actors	Administrator user
Pre-conditions	• Administrator have to be logged in.
Inputs	• Specific commands in the user interface
Events sequence	N/A
Extension Points	N/A
Outputs	• Visible feedback in the user interface.
Post-conditions	N/A
Requirements map	N/A
Related Mock up	N/A

Use Case: CS-U09	**Name: Configure Data Archiving Engine**
Description	The Data Archiving Engine will contain a policy how long data will be stored and performs changes. It can be managed by administrators.
Actors	Administrator
Pre-conditions	• The Administrator has to be logged in • The Administrator needs the specific rights
Inputs	• Bucket Identifier • Store Policy
Events sequence	N/A
Extension Points	N/A
Outputs	N/A
Post-conditions	N/A
Requirements map	F20
Related Mock up	N/A

5.3.3 Component Interaction

The Cloud Storage will be used by all other components to store and provide data. Every component can create several types of buckets ((semi-)structured, binary, semantic) that can be used to store data. The cloud based data storage will support a basic set of CRUD operations for all buckets, e.g. for receiving an element based on its ID. Additionally some buckets will provide an advanced set of queries in a suitable data format as for example OData for (semi-)structured data or SparQL for semantic data. This query could be for example a SQL query with a where clause for a SQL bucket.

As part of the authentication system the Cloud Storage will manage a simple access control list which can be queried. The whole communication will be realized via the Message Routing component. This can be seen in Figure 59, the component that has created the bucket can access it automatically. To make it reachable for other components it is necessary to set adequate rights. This will be normally be performed by an Administrator.

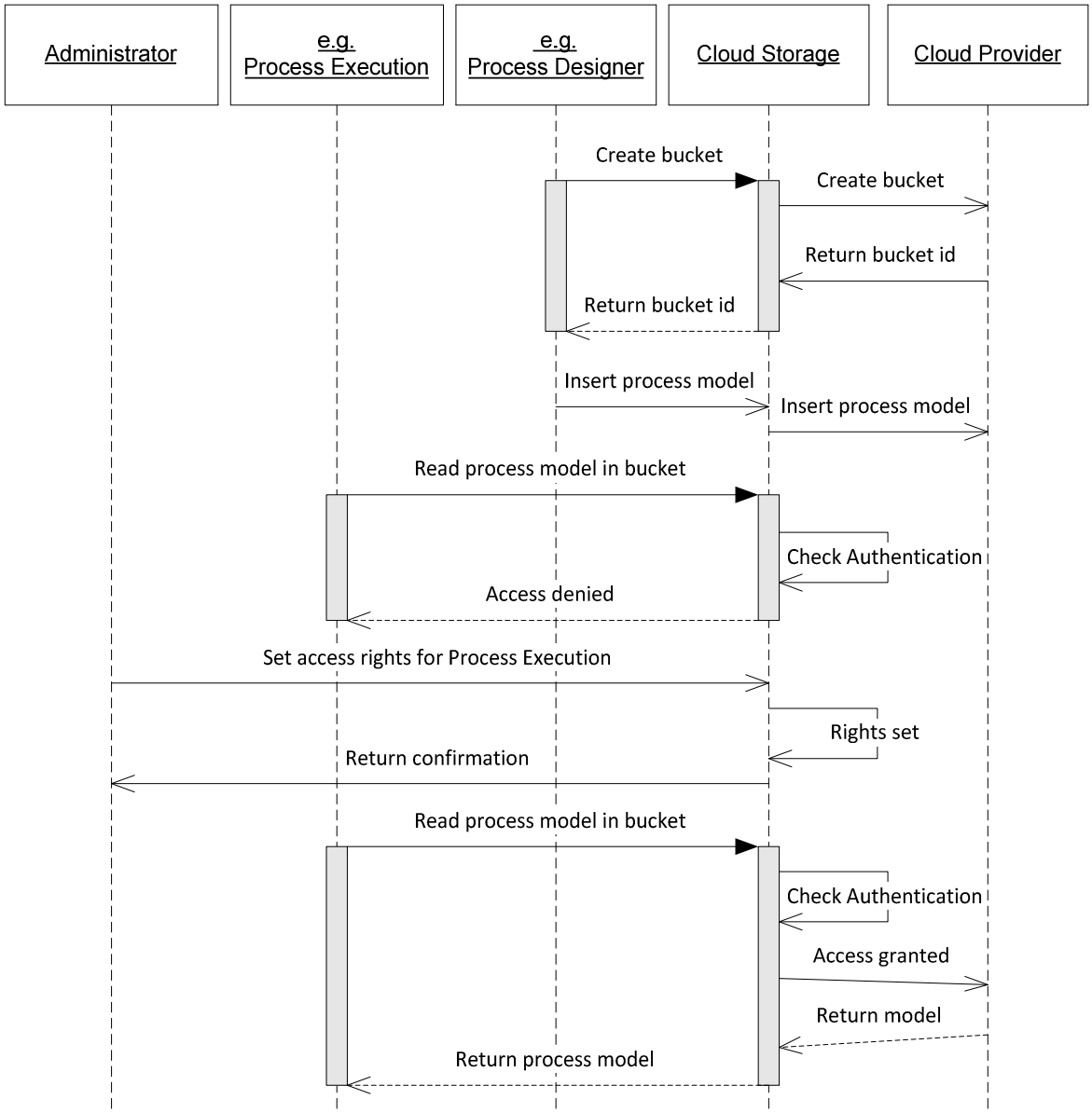

Figure 38 - Access List Functionality in the Cloud Storage

5.3.4 Conceptual Data Model

The Cloud Storage will provide basic CRUD requests for all bucket types. For requests on (semi-)structured Databases a simple protocol with an open format, like OData or GData, will be used. For semantic buckets the Cloud Storage will use a RDF query language such as SparQL.

The stored data structures have to be defined by the components that will store the data.

5.3.5 Parameters to Take Into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+++
Regularly Updated	+
Technical Up-to-Dateness / Appeal	+
Open Source	+/-
Non-Infecting	++
Code-Quality	+
Extensibility	+
Community	+
Performance	+++
Reuse of existing developments	++
EU project origin	+/-
Platform (Portability)	++
Open Standards Compliance	+
Interoperability	++
Specific Parameters	
Scalability	+
ACID compliant	+++
Redundant data storage	++
CRUD operations	+++
Backup	-
Replications	-
Relations / References	++
Transactions	+/-
Costs for data storage	+
Version control (Binary storage)	+
Large file support (Binary storage)	++
Tagging (Binary storage)	+/-
Key/Value storage ((Semi-)Structured Storage)	+++
Schema less ((Semi-)Structured Storage)	+++
SparQL support (Semantic storage)	+++

5.4 Data Provisioning and Discovery

The *Data Provisioning and Discovery* (DPD) module provides a model and tools for representation, publication and discovery of the key artifacts of the ADVENTURE, required for the design, management and operation of Virtual Factories.

5.4.1 Overall Functional Characterization

In order to be able to provide the Virtual Factory functionality, ADVENTURE requires appropriate data definition and maintenance. Since many different independent parties are involved in the virtual manufacturing, ADVENTURE faces the challenge of integrating various data types from different origins and sources. In this regard efficient Data Provisioning and Discovery is of major importance and constitutes a substantial part of the ADVENTURE research and development.

The first step of *joining* the ADVENTURE is mainly related to Data Provisioning. The Data Provisioning component defines formal, structured, semantic-driven descriptions of factories, their products, their demands and offers in terms of semantically enriched services and their properties. Examples of such properties are skills, locations, standards, regulations, capabilities, current capacities, availabilities, cost, delivery time, carbon footprint, etc. These descriptions are provisioned through the ADVENTURE semantic infrastructure and can be correspondingly found and compared and eventually involved into production processes by the interested parties at the second step - *searching*. This will ensure that factories can interoperate at technical level and that they can be involved in complex processes based on their abilities to offer competitive services in the next steps of *plug* and *play*.

Along with the data description, Data Provisioning and Discovery also includes the mechanisms that act upon these data descriptions, that is, for provisioning (or deployment on the platform) and discovery (or search inside the platform, e.g., find partners and partner services). These mechanisms may concern different aspects, such as business activities, manufacturing processes, companies, factories, services, offers/demands, etc.

Following is a description of the ADVENTURE functionalities constituting the corresponding Data Provisioning and Discovery Support in the form of Use Cases and related diagrams.

5.4.2 Data Provisioning Use Cases

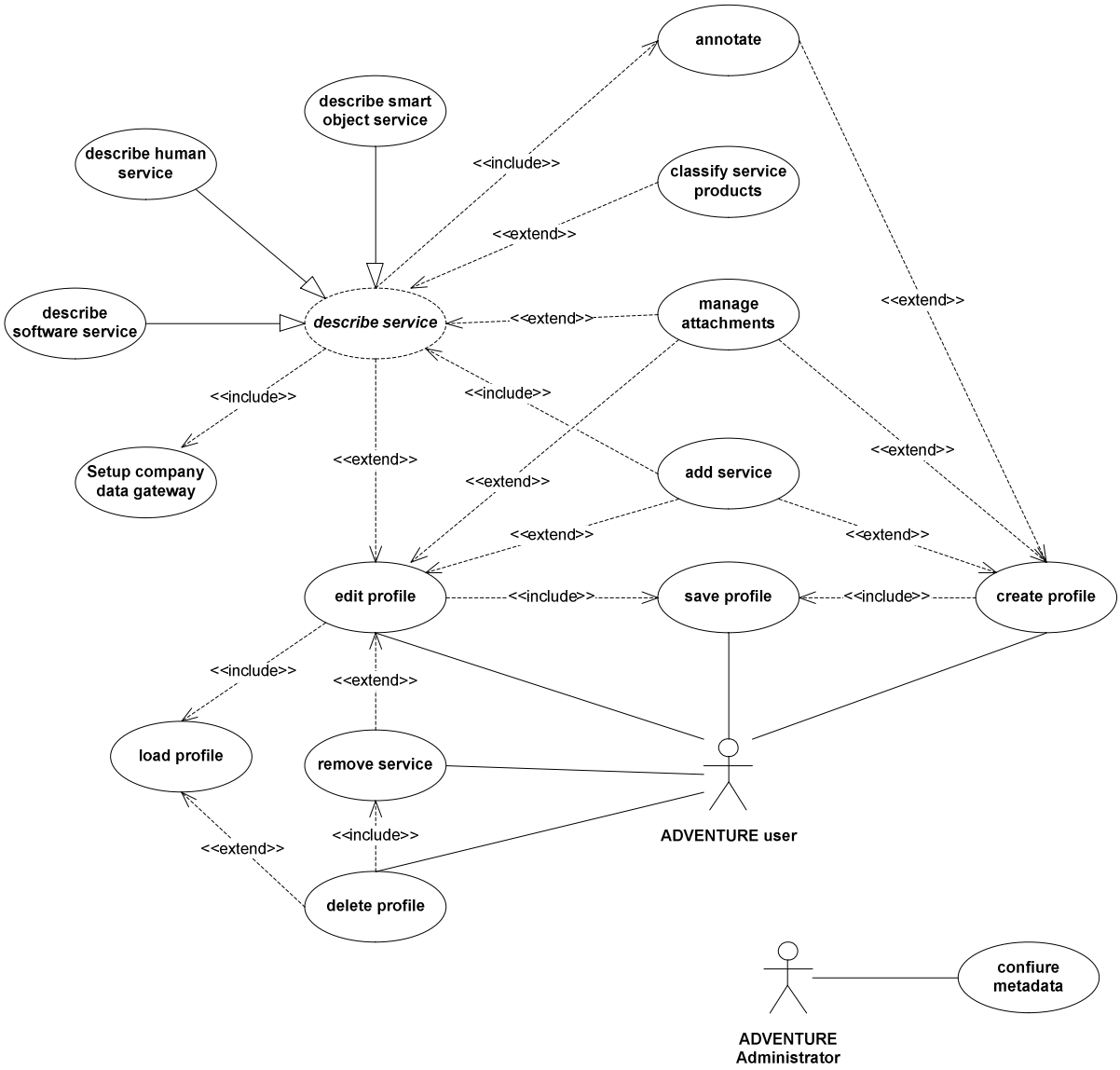

Figure 39 - Data Provisioning Use Cases

Use Case: DP-U01	Name: Create Profile
Description	An ADVENTURE user who wants to involve an organization in Virtual Factories creates a profile for that organization.
Actors	ADVENTURE User
Pre-conditions	• The ADVENTURE user is authenticated and authorized for the operation
Inputs	• Actor identity
Events sequence	• The actor invokes “create profile” functionality on the

	Dashboard
	• The system displays an empty profile form in the profile editor
	• The user fills in all the information requested within the form
	• The profile is ready for saving:
	  • Default: The user saves the profile: Include DP-U02 “Save Profile” Use Case.
	  • The user may cancel the process at any time before saving:
	  • The actor is presented with the profile editor in initial state.
	  • All changes are discarded

Extension points	• Use Case DP-U04 “Manage attachments” • Use Case DP-U06 “Add Service” • Use Case DP-U11 “Annotate”
Outputs	• New profile
Post-conditions	The created profile is available in the ADVENTURE system.
Requirements map	F1, F4
Related Mock up	Figure 40 - Profile Editor

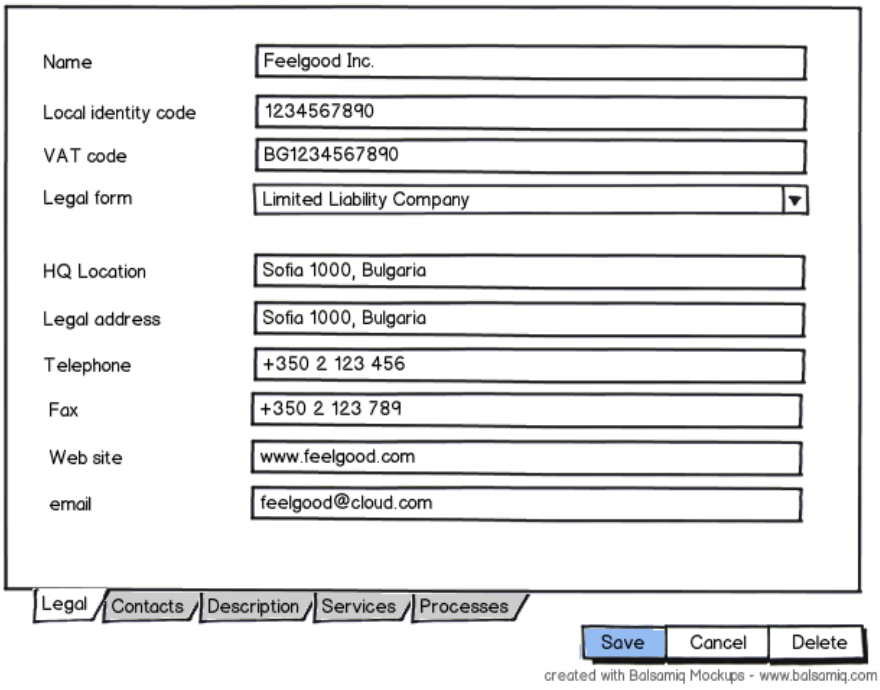

Figure 40 - Profile Editor Overview

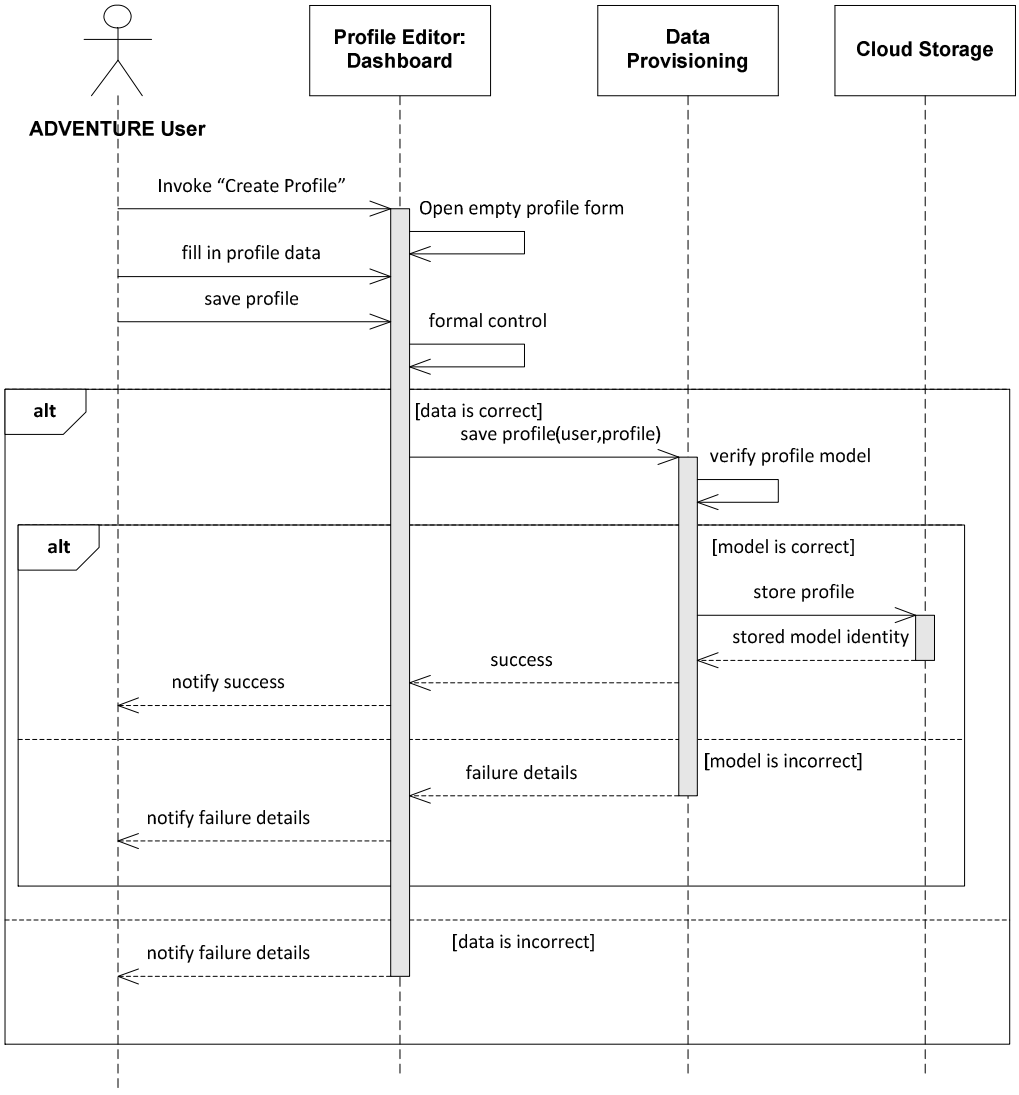

Figure 41 - Create Profile Sequence Diagram

Use Case: DP-U02	**Name: Save Profile**
Description	An ADVENTURE user saves an organization profile after creating a new one or editing an existing one.
Actors	ADVENTURE user
Pre-conditions	• The user is authenticated and authorized for the operation. • The user should have filled in the profiles mandatory data.
Inputs	• ADVENTURE member profile model (with identity if it's an existing one)
Events sequence	• The system verifies the provided model for correctness and consistency

	• Default: The profile model is verified successfully
	  • Default: The system stores the profile
	  • The system displays an information message
	  • An unexpected problem occurred (e.g. vital system component is not available too long or the profile assets could not be stored as consistent, atomic entities
	  • Changes to the system, if any, are reverted to pre-save state
	  • The system displays an error message
	• The profile is saved and not published yet
	  • The system provides the actor with option to publish the profile
	  • The user chooses to publish the profile
	  • The profile is published, discoverable and can be used in Virtual Factory compositions.
	• There is a problem with the correctness or consistency in the profile model
	  • Changes to the system, if any, are reverted to pre-save state
	  • The system displays an error message

Extension points	N/A
Outputs	• ADVENTURE member profile identity
Post-conditions	The created member profile model is available in the ADVENTURE system, i.e. can be discovered.
Requirements map	F1, F10, F84
Related Mock up	Figure 40 - Profile Editor

Use Case: DP-U03	**Name: Classify Service Products**
Description	An ADVENTURE member uses predefined taxonomies to classify service resources, requirements and productions in a common model, so that matching services can be found.
Actors	ADVENTURE member (user)
Pre-conditions	• The user is authenticated and authorized for the operation.
Inputs	• Product Taxonomy, service
Events sequence	• The user loads product taxonomy • The user browses product taxonomy • The user selects a category • The user annotates service resource/product with the selected category: Includes DP-U11 “Annotate” Use Case • Default: The service model is updated. • The user may cancel the process at any time before saving: • The user is presented with the profile editor in initial

	state. • All changes are discarded
Extension points	N/A
Outputs	• Annotated service
Post-conditions	N/A
Requirements map	F1, F10, F84
Related Mock up	Figure 48 - Service Annotation Figure 49 - Service Annotation Concept Selection

Use Case: DP-U04	**Name: Manage Attachments**
Description	ADVENTURE member adds or removes attachments to their profile, processes or services (this might be certificates, images of the products, specifications, etc.)
Actors	ADVENTURE member (user)
Pre-conditions	• The user is authenticated and authorized for the operation
Inputs	• Containing element identity

Events sequence	• The user selects element (profile, service, process) to add/remove attachment to/from
	• Default: The user invokes add attachment
	• The user invokes remove attachment
	  • The system retrieves all the attachments for this element
	  • The system displays a list of the attachments
	  • The user chooses an attachment to remove
	  • The system removes the attachment
	• The system opens the add attachment form
	• The system opens the browse dialogue
	• The user chooses attachment from their system
	• The system loads the attachment
	• The system checks the attachment
	• Default: The attachment is OK
	  • The system saves the attachment.
	• The attachment is invalid
	  • An error message is displayed

Extension points	N/A
Outputs	• Attachment saved and linked to an element
Post-conditions	N/A
Requirements map	F65
Related Mock up	N/A

Use Case: DP-U05	Name: Edit Profile
Description	ADVENTURE member edits organization profile
Actors	ADVENTURE member (user)
Pre-conditions	• The user is authenticated and authorized for the operation • There is an existing profile that this actor is eligible to edit.
Inputs	• Profile

Events sequence	• The user invokes edit profile functionality
	• The system opens the profile editor and displays the current data: Include DP-U07 “Load profile” Use case
	• The user edits the data
	• The user saves the profile: Include DP-U02 “Save Profile” Use Case
	• The user may cancel the process at any time before saving:
	  • The user is presented with the profile editor in initial state.
	  • All changes are discarded

Extension points	• Use Case DP-U04 “Manage Attachment”, • Use Case DP-U06 “Add Service”, • Use Case DP-U08 “Describe Service” • Use Case DP-U10 “Remove Service”
Outputs	• Success/error notification
Post-conditions	• Notifications about the changes in the profile are sent to respective partners in collaboration (suppliers or customers)
Requirements map	F1, F10, F84
Related Mock up	Figure 40 - Profile Editor

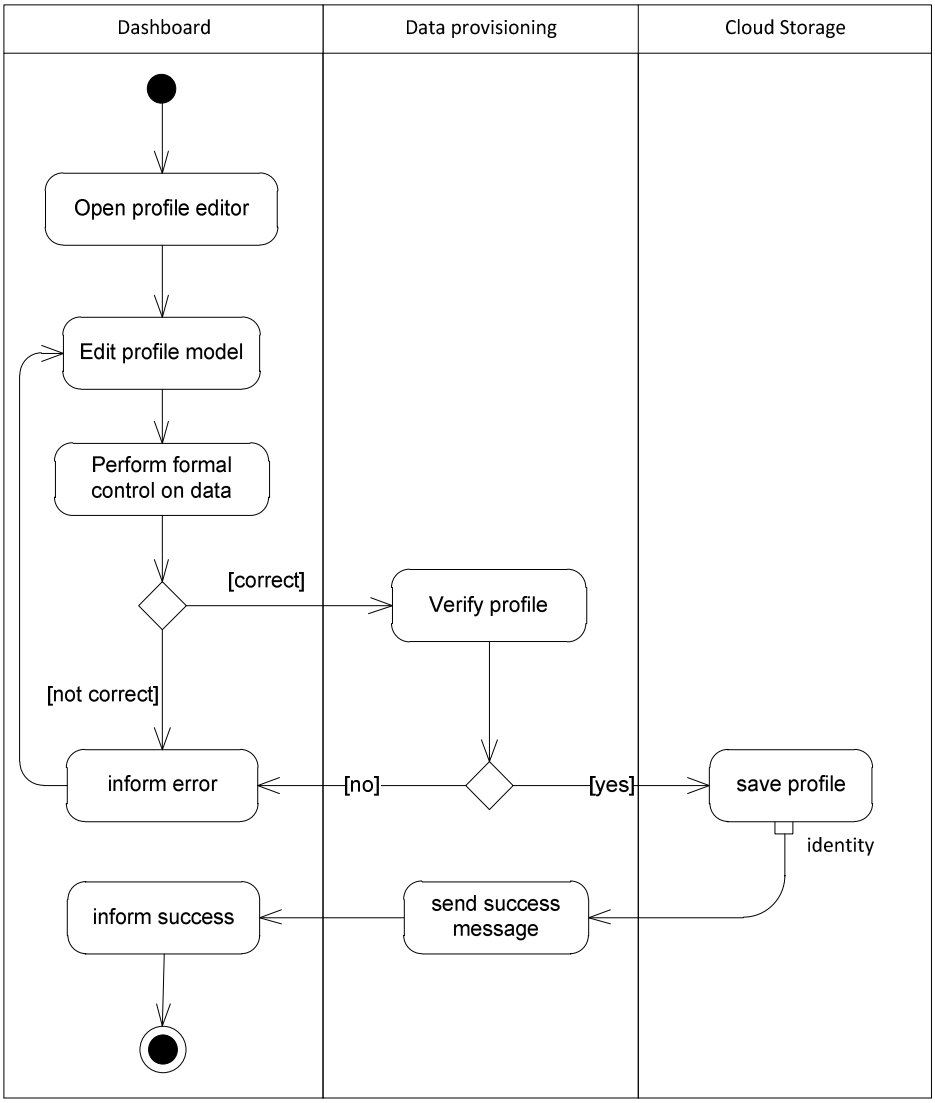

Figure 42 - Edit Profile Activity Diagram

Use Case: DP-U06	**Name: Add Services**
Description	ADVENTURE member adds services to their organization profile (this might be human service, smart object service or software service)
Actors	ADVENTURE member (user)
Pre-conditions	• The user is authenticated and authorized for the operation • There is a profile created to which services will be added.
Inputs	N/A
Events sequence	• The user invokes “add service” functionality • The user selects the type of service to add • Default: The system opens the smart object service

	description form: Includes “Describe Smart Object Service” Use Case.
	• The system opens the human service description form: Includes “Describe Human Service” Use Case.
	• The system opens the software service description form: Includes “Describe Software Service” Use Case
	• The user describes the service: Include DP-U08 “Describe Service” Use Case
	• Default: The user saves the profile: Include DP-U02 “Save Profile” Use Case
	• The user may cancel the process at any time before saving:
	  • The user is presented with the profile editor in initial state.
	  • All changes are discarded

Extension points	N/A
Outputs	• Service described and added to profile
Post-conditions	N/A
Requirements map	F77, F78, F85
Related Mock up	N/A

Use Case: DP-U07	**Name: Load Profile**
Description	ADVENTURE Member wants to load an existing profile
Actors	ADVENTURE Member
Pre-conditions	• The ADVENTURE Member is authenticated and authorized for the operation • Profile is available and has relevant user has permissions
Inputs	• Actor identity

Events sequence	• The ADVENTURE user invokes load profile functionality by selecting a profile from a list in the Dashboard
	• The system retrieves all the static information related to this profile (including attachments).
	• The system checks whether the services are involved in process design or in a process in execution
	• Default: The services are not involved in process design or in a process in execution
	  • The system retrieves information about the services related to this profile.
	  • The system displays the profile information together with a list of related services in the profile editor.
	• The services are involved in process that is in design
	  • The system retrieves information about the process.
	  • The system displays the information about the process.
	• The services are involved in process that is in

	execution • The system retrieves all the dynamic information for the service status. • The system displays all the dynamic information for the service status.
Extension points	N/A
Outputs	• Profile and its services are is displayed for further use
Post-conditions	N/A
Requirements map	F1, F3, F4, F11
Related Mock up	Figure 40 - Profile Editor

Use Case: DP-U08	**Name: Describe Service**
Description	ADVENTURE member describes services (this might be human service, smart object service or software service)
Actors	ADVENTURE member
Pre-conditions	• The ADVENTURE user is authenticated and authorized for the operation • There are services created that will be described.
Inputs	• Service type
Events sequence	• The ADVENTURE user invokes the service description functionality • Default: The system opens the software service description form • The system opens the smart object service description form • The system opens the human service description form • The user fills in the service general information • The user annotates the service: Includes DP-U11 “Annotate” Use Case • The user setups the service gateway: Includes Use Case GW-U05 “Setup company data gateway” • The system updates the service model
Extension points	• Use Case DP-U04 “Manage Attachment”, • Use Case DP-U03 “Classify service products”
Outputs	• Service model
Post-conditions	N/A
Requirements map	F8, F10, F11
Related Mock up	Figure 43 - Describe Service Figure 44 - Describe Service UI Figure 45 - Describe Service

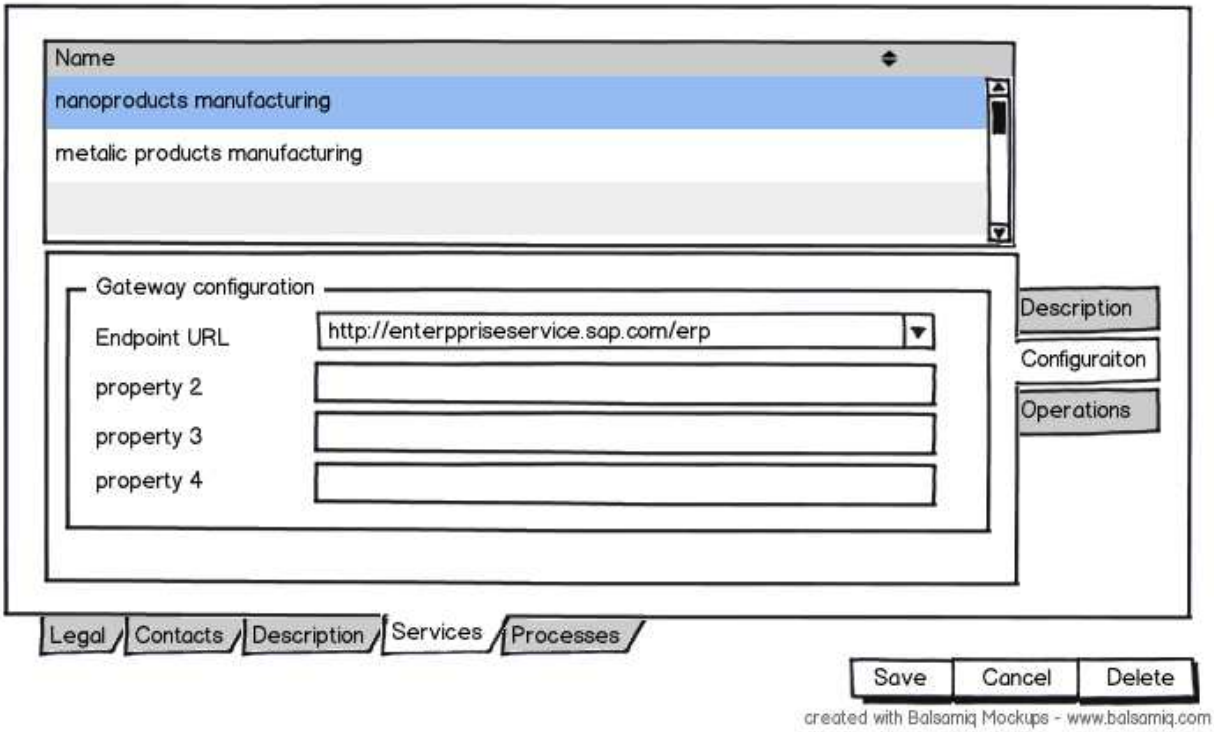

Figure 43 - Describe Service Configuration UI

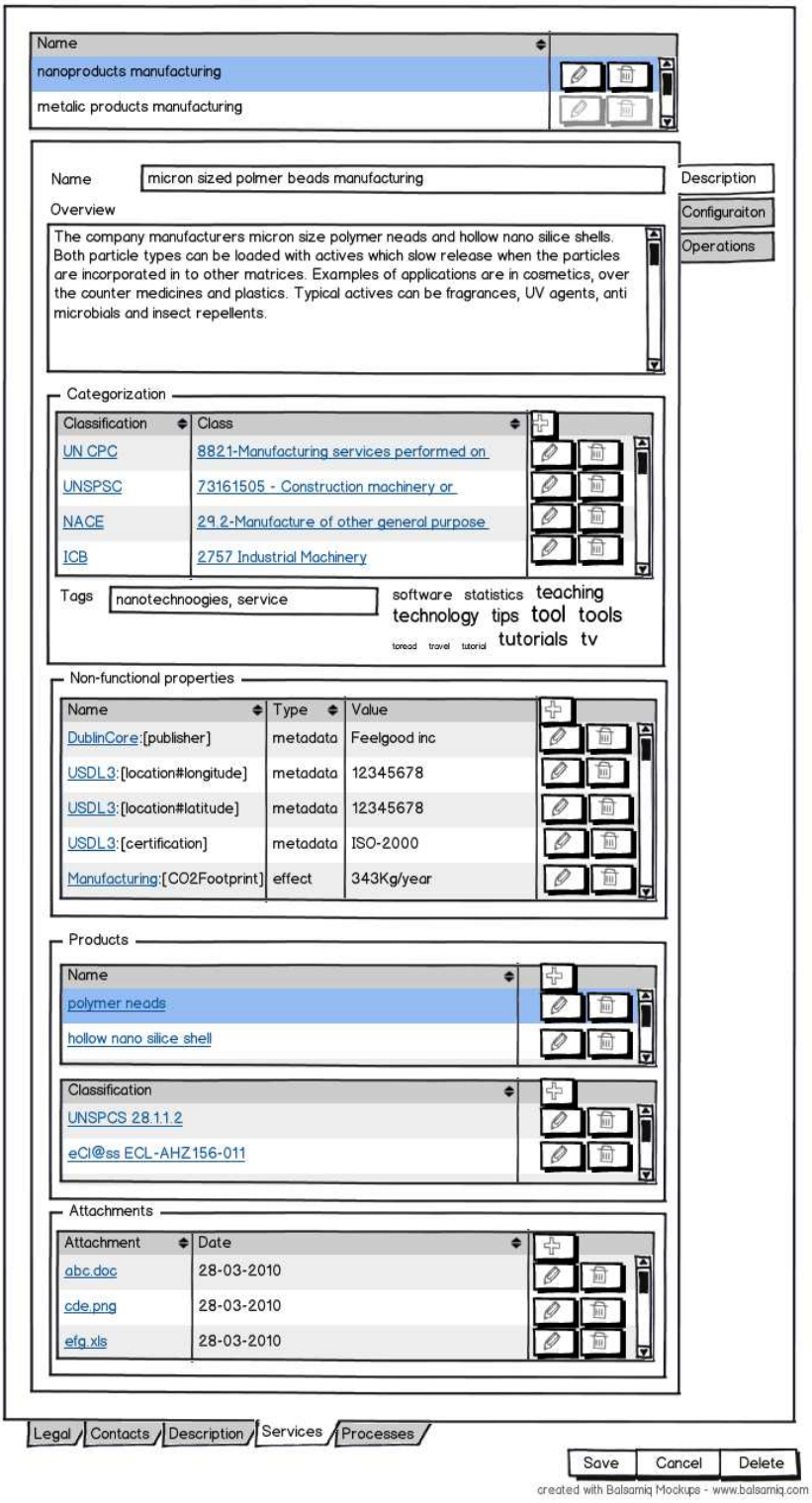

Figure 44 - Describe Service UI

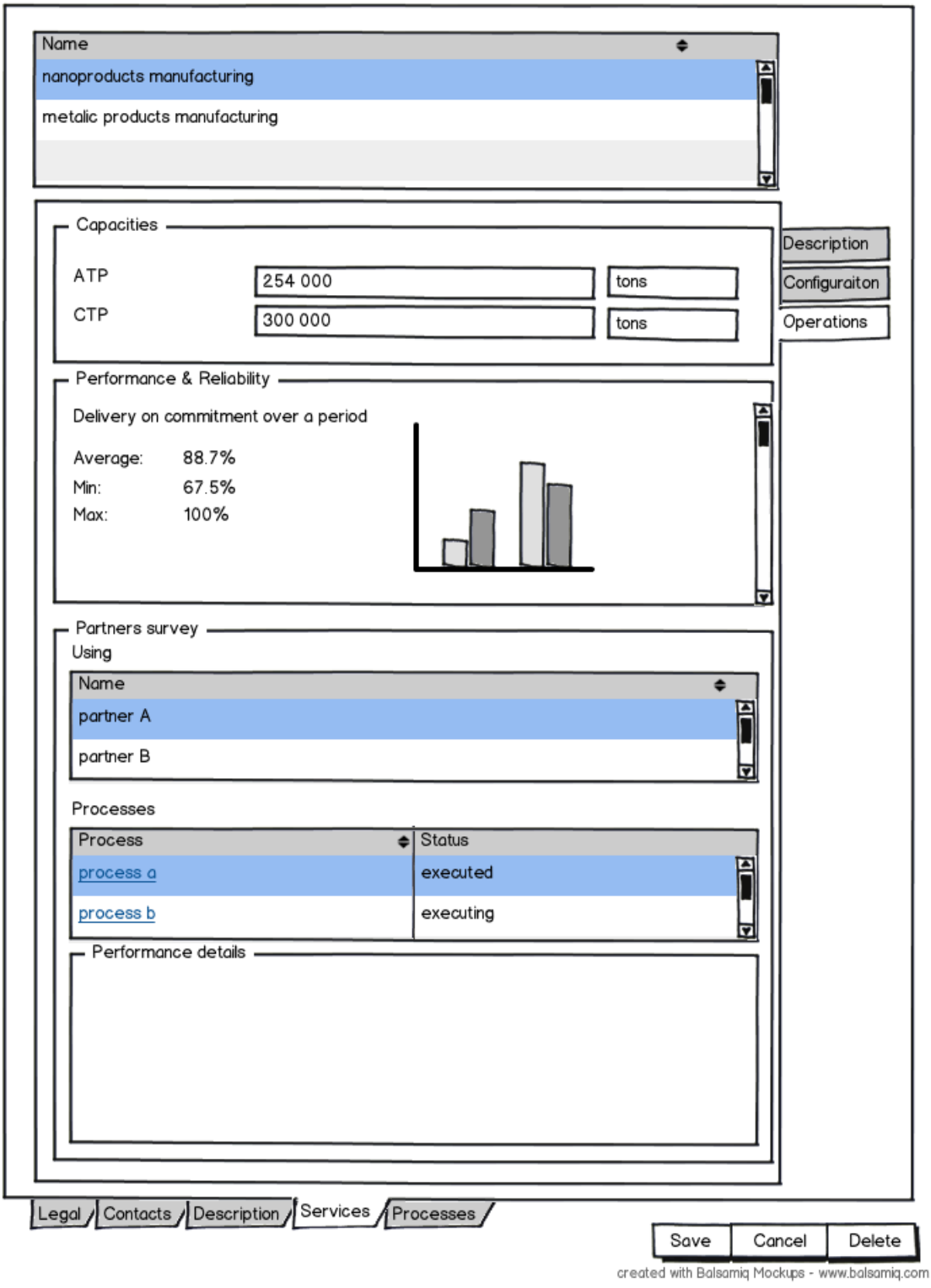

Figure 45 - Describe Service Operations UI

Use Case: DP-U09	Name: Delete Profile
Description	ADVENTURE member deletes their organization profile
Actors	ADVENTURE member (user)
Pre-conditions	• The user is authenticated and authorized for the operation • There is a profile created which will be deleted
Inputs	Actor identity, profile identity

Events sequence	• The user invokes the delete profile functionality by selecting a profile from a list in the Dashboard
	• The system requests confirmation from the actor
	• The user confirms the deletion.
	• The system checks whether the services linked to the profile are bound to activities in other process designs or are in execution
	• Default: Profile services are not bound to activities in other process designs
	  • The system deletes the profile: Includes Use Case DP-U10 “Remove service”
	• At least one service linked to the profile is bound to activities in other process designs
	  • The system notifies the actor and requests confirmation
	  • The user confirms the deletion
	  • The system deactivates the profile
	  • The system notifies the respective brokers that the bindings are not consistent any more.
	• At least one service linked to the profile is involved in a process instance in execution state
	  • The system notifies the user that the service cannot be removed
	  • The system displays a list of respective Virtual Factories’ brokers to be contacted

Extension points	• Use Case DP-U07 “Load profile”
Outputs	• Success/failure notification.
Post-conditions	N/A
Requirements map	F1, F3, F4
Related Mock up	Figure 40 - Profile Editor

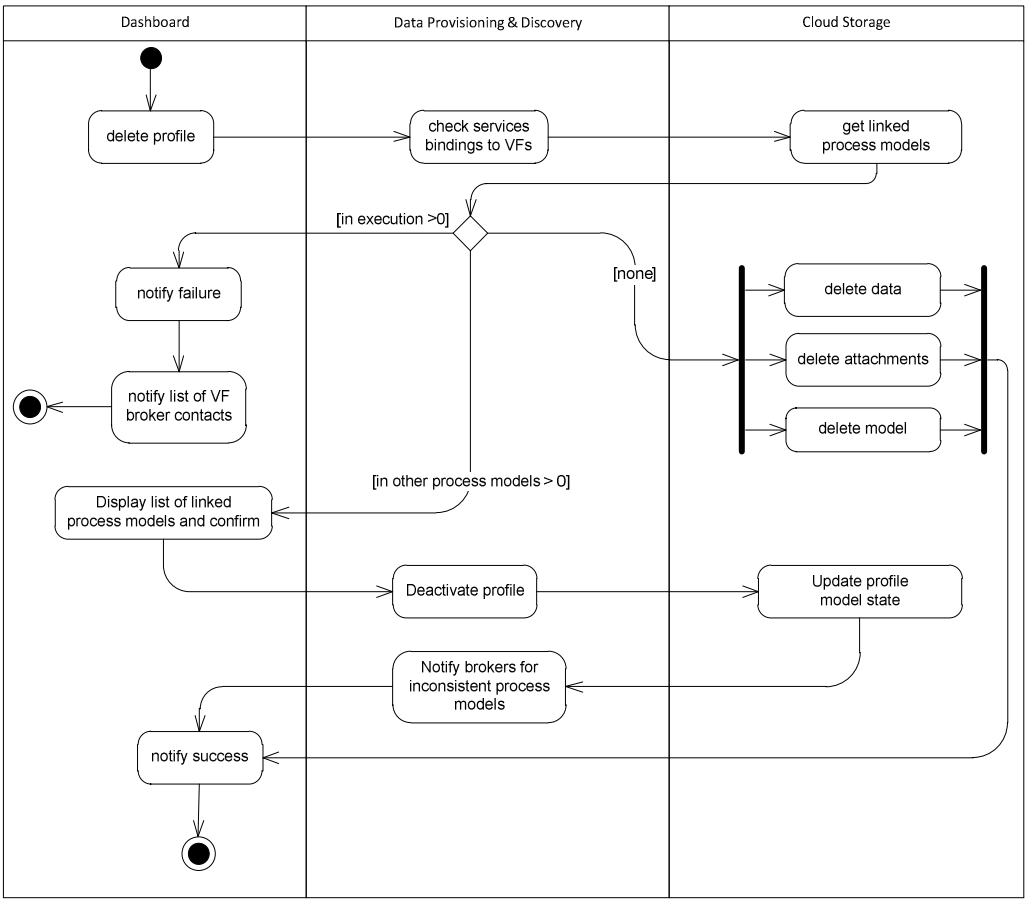

Figure 46 - Delete Profile Activities

Use Case: DP-U10	**Name: Remove service**
Description	ADVENTURE member removes a service from organization profile (this might be human service, smart object service or software service)
Actors	ADVENTURE member (user)
Pre-conditions	• The ADVENTURE member is authenticated and authorized for the operation • There is a service created which will be removed
Inputs	• User identity, service identity

Events sequence	• The user invokes remove service functionality by selecting a service from a list in the Dashboard
	• The system checks whether the service is bound to activities in other process designs or is in execution
	• Default: The service is not bound to activities in other process designs
	  • The system requests confirmation from the user
	  • The user confirms the removal.
	  • The system removes the service.

	• The service is bound to activities in other process designs that are not executed yet.
	  • The system notifies the user
	  • The system requests confirmation from the user.
	  • The user confirms the removal
	  • The system removes the service
	  • The system notifies the respective users(brokers) that the bindings are not consistent any more
	• The service is involved in a process instance that is in execution
	  • The system notifies the user that the service cannot be removed
	  • The system displays a list of Virtual Factories to be contacted
	• The service was involved in an already executed and completed process instance
	  • The system notifies the user
	  • The system requests confirmation from the user.
	  • The user confirms deactivation
	  • The system deactivates the service
	  • The system notifies the respective users(brokers) that the bindings are not consistent any more

Extension points	• Use Case DP-U04 “Manage Attachment”, • Use Case DP-U03 “Classify service products”
Outputs	• Service removed
Post-conditions	N/A
Requirements map	F1, F3, F4
Related Mock up	Figure 40 - Profile Editor

Use Case: DP-U11	**Name: Annotate**
Description	ADVENTURE member annotates their profile, services, processes, process elements (using ontologies existing in ADVENTURE) in order to make them visible and searchable within the system.
Actors	• ADVENTURE member
Pre-conditions	• The ADVENTURE user is authenticated and authorized for the operation • Relevant ontologies/taxonomies/classifications are available in ADVENTURE.
Inputs	• Profile or service or process or process step to be annotated
Events sequence	• The ADVENTURE user invokes annotation functionality. • The system provides the actor with the relevant to the annotated entity type ontologies/ taxonomies/classifications. • The user links their data to ontology elements. • Default: The system saves the links

	• The system displays a confirmation message • The actor may cancel the process at any time before saving. • The actor is presented with the profile editor in initial state. • All annotations are discarded
Extension points	N/A
Outputs	• Annotated profile, service, process or process element
Post-conditions	• The profile, service, process or process element has associated metadata that is indexed and can be searched and found in ADVENTURE
Requirements map	F11, F76, F77
Related Mock up	Figure 48 - Service Annotation Figure 49 - Service Annotation Concept

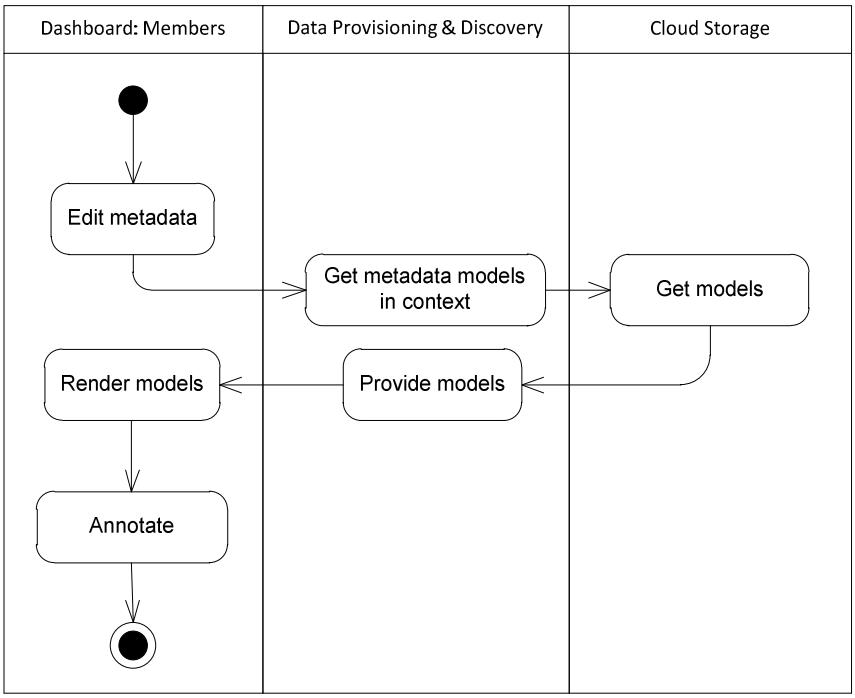

Figure 47 - Annotate Activity Diagram

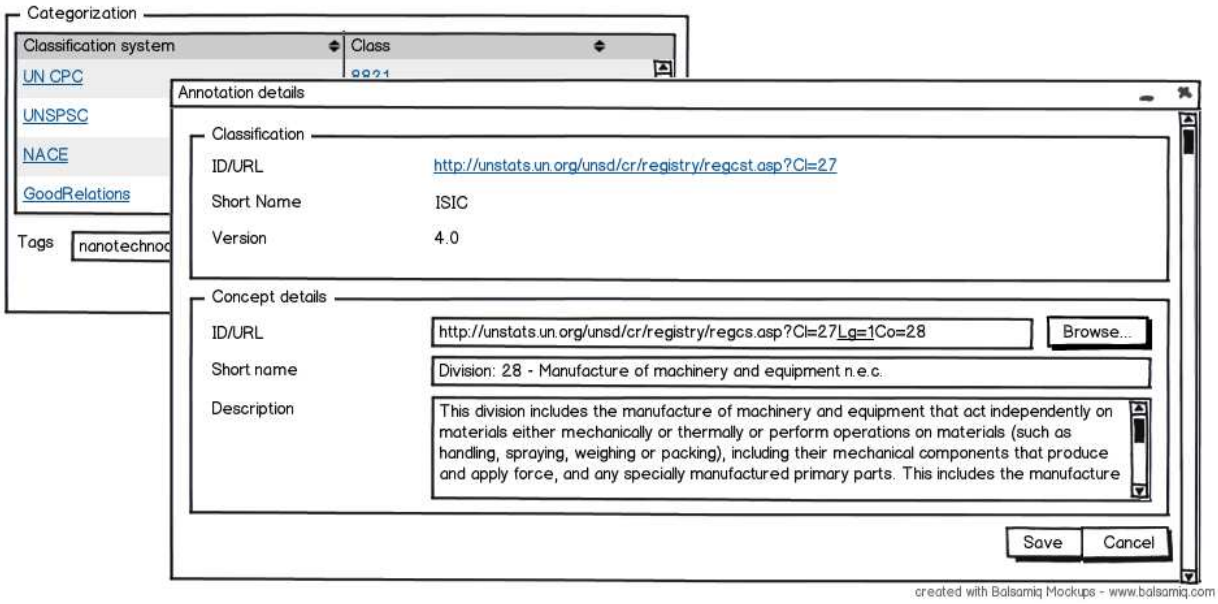

Figure 48 - Service Annotation Dialogue

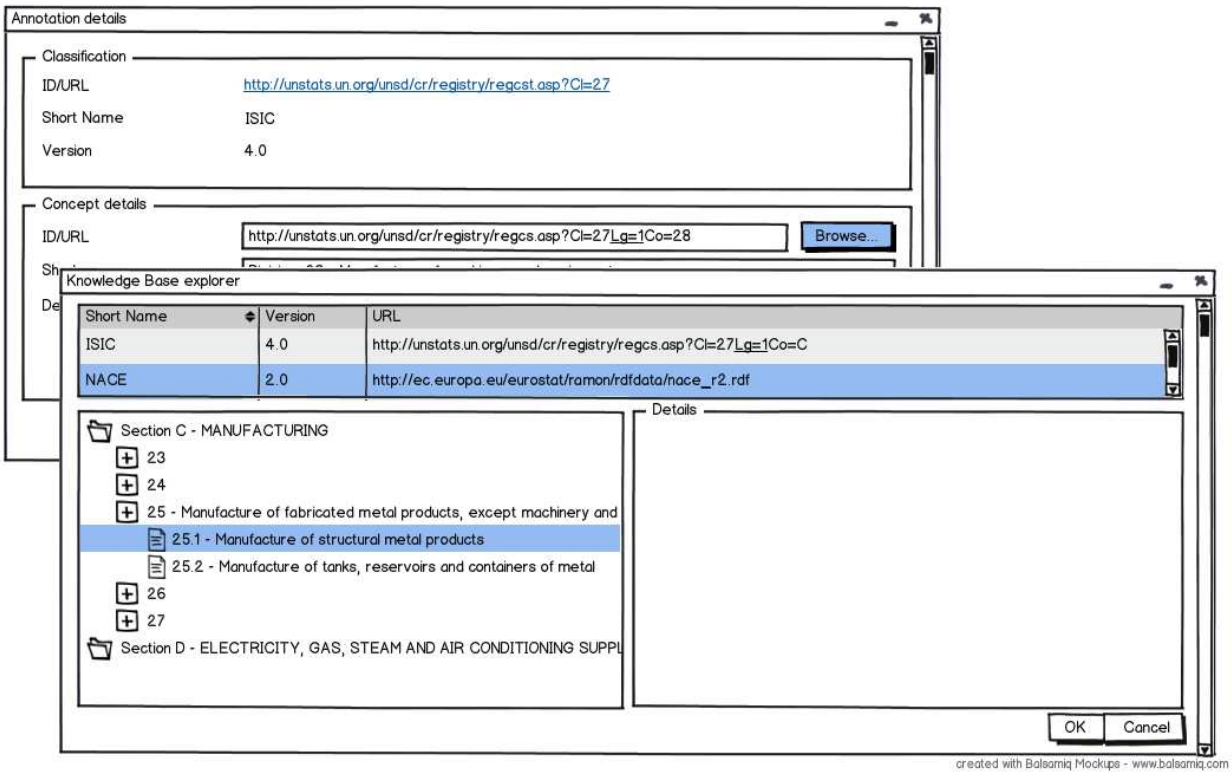

Figure 49 - Service Annotation Concept Selection Dialogue

Use Case: DP-U12	Name: Configure Metadata
Description	ADVENTURE administrator configures the Data Provisioning and Discovery component to use particular metadata structures (ontologies/taxonomies/classifications), which will be shared in the system for annotation and search in different contexts.
Actors	ADVENTURE administrator
Pre-conditions	• The user is authenticated and authorized for the operation.
Inputs	• Metadata structure

Events sequence	• The ADVENTURE administrator invokes configuration functionality.
	• The system opens the configuration editor.
	• The ADVENTURE administrator defines relevant usage contexts
	• Default: The ADVENTURE administrator uploads metadata model specifications encoded in a set of accepted serialization formats
	  • The system verifies the provided model for correctness (e.g. information space consistency after integration)
	  • Default: The model is correct
	  • The system integrates the metadata model (stores and makes available in DP&D interfaces)
	  • The system displays a confirmation message
	  • The system finds out that the provided model is incompatible
	  • The system sends an error message to the user.
	  • The model is not integrated. The editor preserves its pre-upload state.
	• The user may cancel the process at any time before saving.
	  • The user is presented with the profile editor in initial state.
	  • All changes are discarded

Extension points	N/A
Outputs	• Success/error message
Post-conditions	• The new metadata is integrated in the Data Provisioning and Discovery and made available in all relevant contexts.
Requirements map	F11, F76, F77
Related Mock up	Figure 51 - Configure

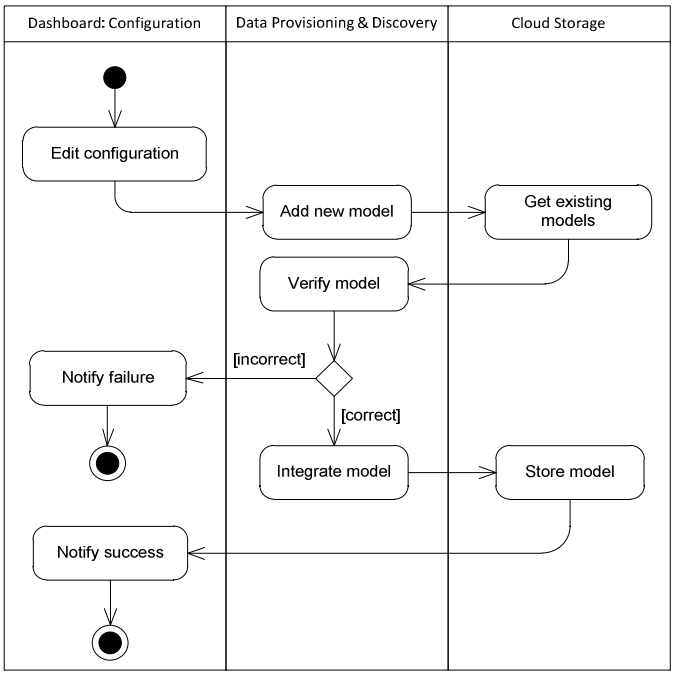

Figure 50 - Configure Metadata Activity Diagram

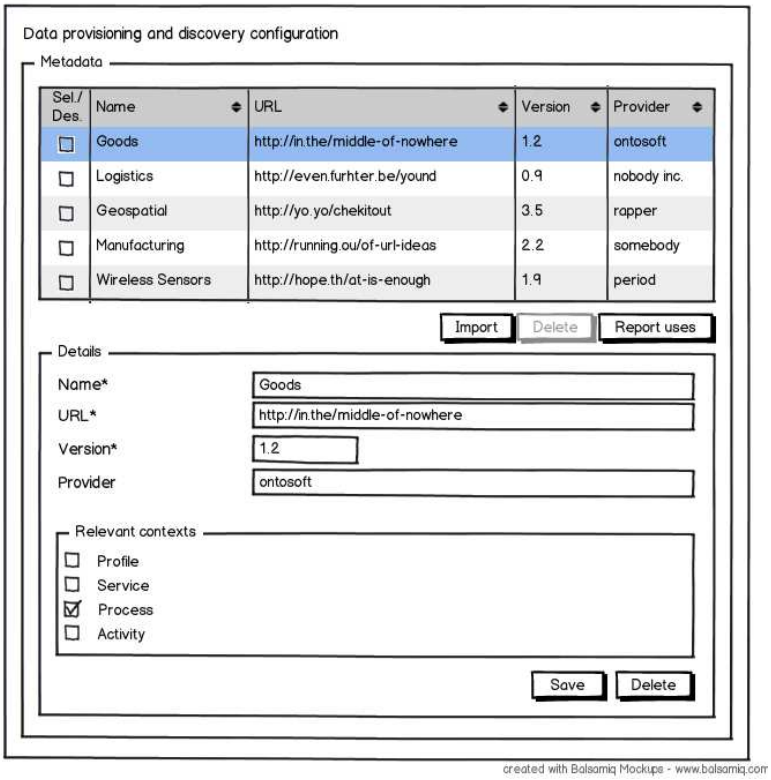

Figure 51 - Configure Metadata UI

5.4.3 Data Discovery Use Cases

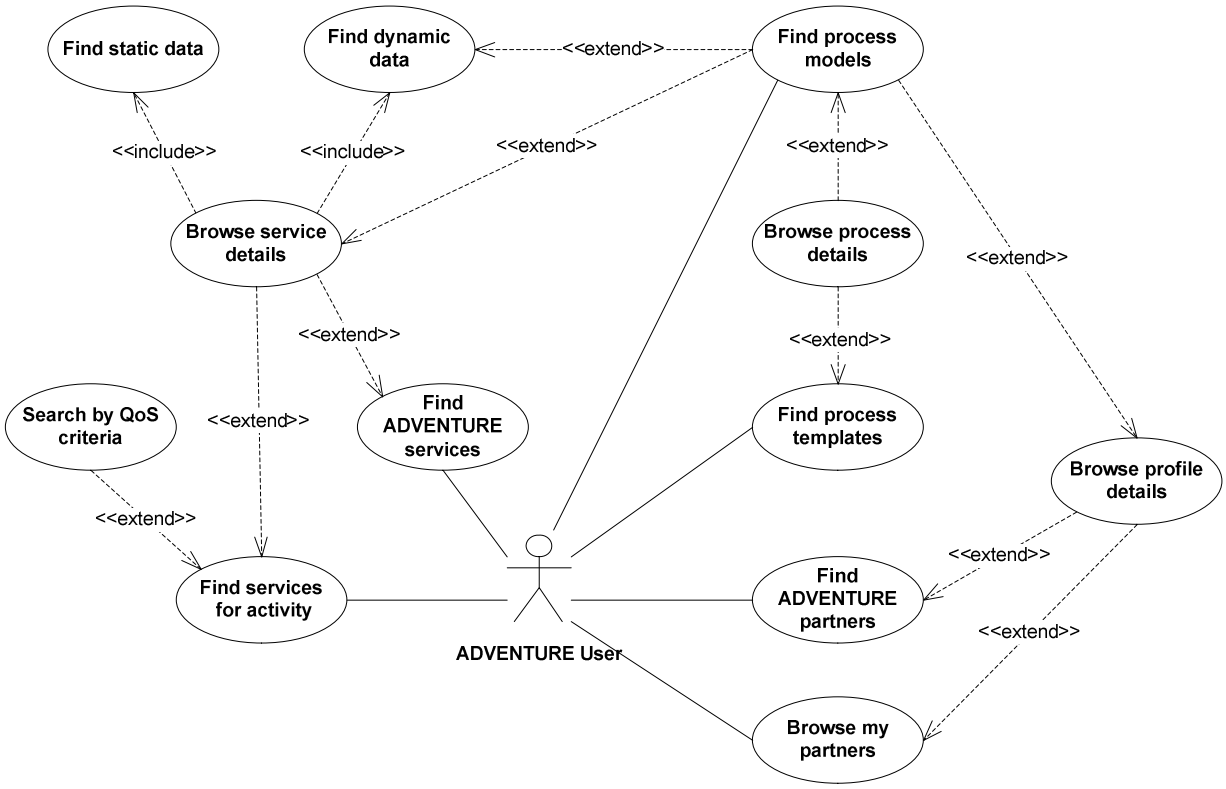

Figure 52 - Data Discovery Use Cases

Use Case: DD-U01	Name: Browse my partners
Description	ADVENTURE member who has already been involved in Virtual Factories (as a customer or as a supplier) browses their partners.
Actors	ADVENTURE member
Pre-conditions	• The user is authenticated and identified as ADVENTURE member.
Inputs	• Actor identity

Events sequence	• Default: The user invokes “browse my partners” functionality
	  • The system retrieves the partners the actor has already been in collaboration with.
	  • The system displays a list of partners that are currently or have been into collaboration with this party (in Virtual Factory).
	• The user filters the search by defining search criteria to narrow the results
	  • The system retrieves the partners that fit the search criteria
	  • The system displays a list of partners that comprises

	the selected search criteria parameters.
Extension points	• Use Case DP-U02 “Browse profile details”,
Outputs	• List of the current partners of this ADVENTURE member that contains generic identification details that orient the user at a glance, such as company name and collaboration processes.
Post-conditions	N/A
Requirements map	F14a,F15, F17
Related Mock up	Figure 53 - Search UI

Use Case: DD-U02	**Name: View profile details**
Description	ADVENTURE user views the details of an ADVENTURE member.
Actors	ADVENTURE member or Guest
Pre-conditions	• The member profile exists and is published.
Inputs	• Member identity
Events sequence	• The user invokes browse member details by selecting an ADVENTURE member from a list in the Dashboard. • The system retrieves the member profile and the related information such as services supported • The system displays the ADVENTURE member profile.
Extension points	• Use Case DP-U10 “Find process models”
Outputs	• A detailed view of a member profile.
Post-conditions	N/A
Requirements map	F14a, F1
Related Mock up	N/A

Use Case: DD-U03	**Name: Find ADVENTURE partners**
Description	An ADVENTURE user wants to find partners within ADVENTURE that fit certain criteria. This might be domain, geo-location, service type and any other metadata describing a partner and its services in ADVENTURE.
Actors	ADVENTURE user
Pre-conditions	N/A
Inputs	• Search criteria

Events sequence	• The user opens a search form.
	• The user specifies search criteria choosing from predefined parameters.
	• The user initiates search.
	• Default: The system retrieves ADVENTURE members info that fit the criteria
	  • The system displays a list of results.
	• There are no members that fit the criteria
	  • The system notifies the user that no appropriate ADVENTURE members have been found.

Extension points	• Use Case DP-U02 “Browse profile details”

Outputs	• The provided list contains generic identification details that orient the user at a glance and comprise the selected search criteria parameters.
Post-conditions	N/A
Requirements map	F14a, F1
Related Mock up	Figure 53 - Search UI

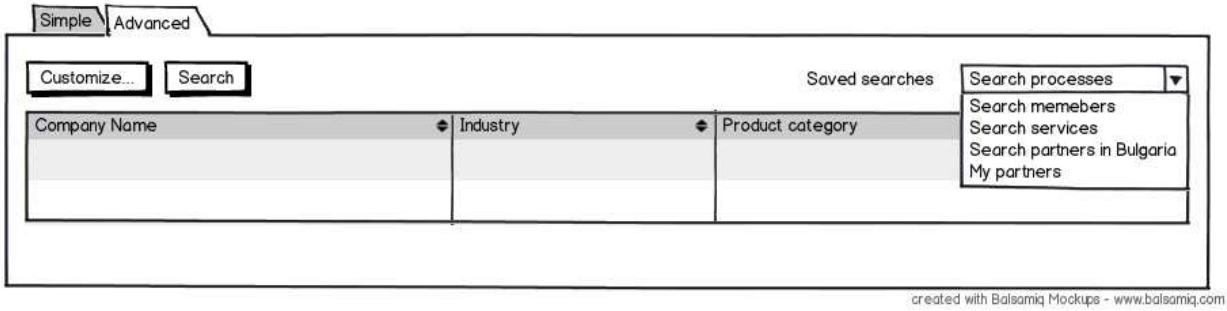

Figure 53 - Search UI Overview

Use Case: DD-U04	**Name: Find ADVENTURE services**
Description	ADVENTURE user wants to find services within ADVENTURE that fit certain criteria. This might be domain, geo-location, service type and any other metadata describing a service in ADVENTURE.
Actors	ADVENTURE user
Pre-conditions	N/A
Inputs	• Search criteria

Events sequence	• The user opens a search form.
	• The user specifies search criteria choosing from predefined parameters.
	• The user initiates search.
	• Default: The system retrieves ADVENTURE services that fit the criteria
	  • The system displays a list of services.
	• There are no services that fit the criteria
	  • The system notifies the user that no appropriate services have been found.

Extension points	• Use Case DP-U06 “Browse service details”
Outputs	• The provided list contains generic details that orient the user at a glance: service name, provider name, functional classification, etc.
Post-conditions	N/A
Requirements map	F78, F84
Related Mock up	Figure 53 - Search UI

Use Case: DD-U05	Name: Find service for activity
Description	An ADVENTURE broker wants to find appropriate services to assigning them to activities in his/her process model at design time or at run time (when an adaptation of the process is needed)
Actors	ADVENTURE broker
Pre-conditions	• Some parameters and constraints should be specified during process activity configuration in order to serve as search criteria.
Inputs	• Activity parameters

Events sequence	• The user initiates “find services” functionality
	• Default: The system retrieves services that fit the criteria (activity parameters)
	  • The system displays a list of services.
	• There are no services that fit the criteria
	  • The system notifies the actor that no appropriate services have been found.

Extension points	• Use Case DP-U09 “Search by QoS criteria” • Use Case DP-U06 “Browse service details”
Outputs	• The provided list contains generic details that orient the user at a glance: service name, company name, functionality, etc
Post-conditions	N/A
Requirements map	F78, F84, F38
Related Mock up	N/A

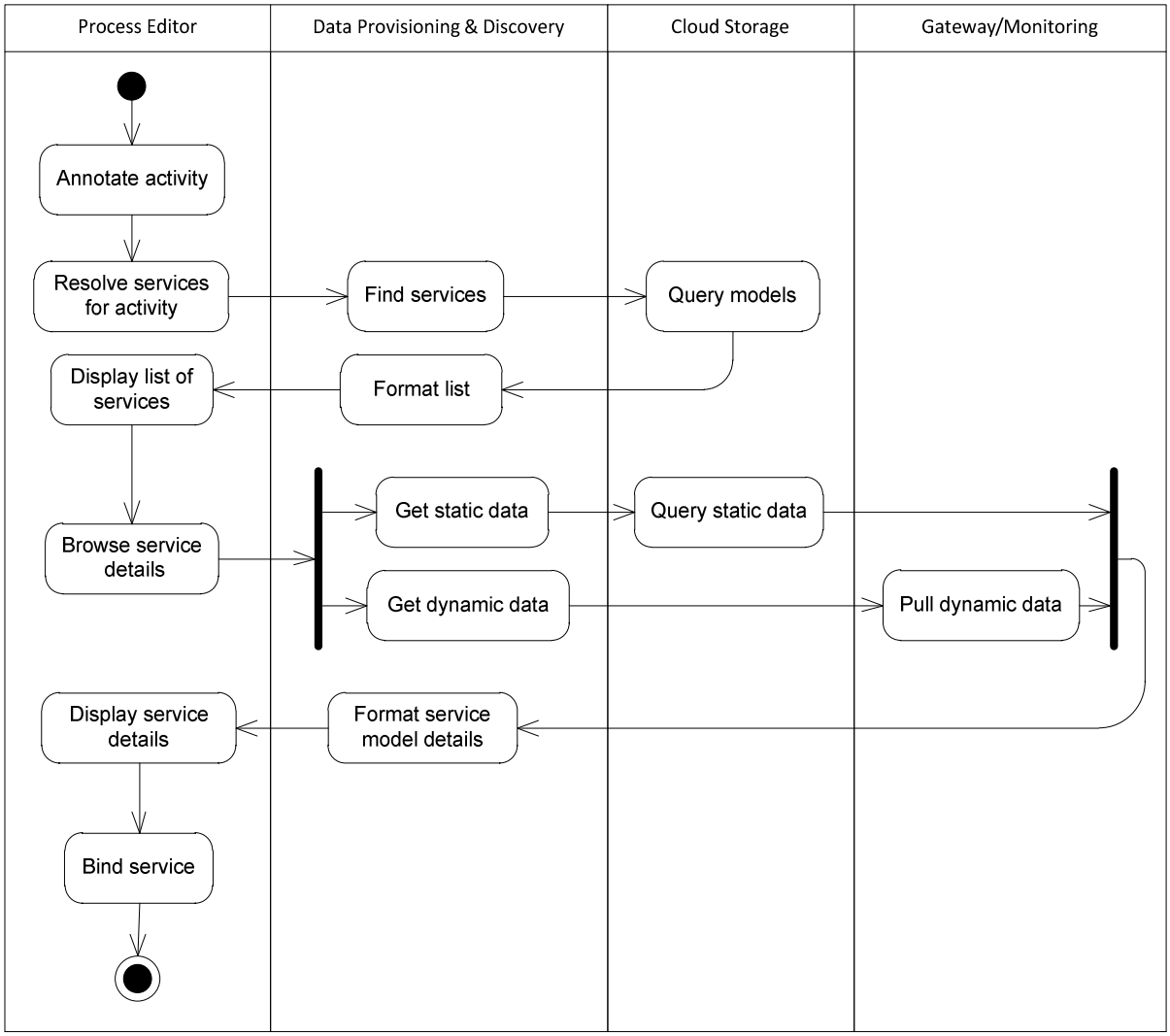

Figure 54 - Find Services for Activity Extended Activity Diagram

Use Case: DD-U06	**Name: Browse service details**
Description	ADVENTURE user browses the details of a service that includes both static and dynamic data
Actors	ADVENTURE member or Guest
Pre-conditions	N/A
Inputs	• Service identity
Events sequence	• The user invokes browse service details by selecting a service from a list. • The system retrieves services static data: Includes Use Case DP-U07 “Find service static data” • The system retrieves service dynamic data: Includes Use Case DP-U08 “Find service dynamic data”. • The system displays the service details.

Extension points	• Use Case DP-U10 “Find process model”
Outputs	• A detailed view of service details.
Post-conditions	N/A
Requirements map	F11, F59, F65
Related Mock up	N/A

Use Case: DD-U07	**Name: Find service static data**
Description	ADVENTURE user wants to see services static data such as name, functionality, company name, etc.
Actors	ADVENTURE user
Pre-conditions	N/A
Inputs	• Service identity
Events sequence	• The user initiates “browse service details” functionality. • The system retrieves services’ static information • The system displays the information
Extension points	N/A
Outputs	• The information contains generic details that orient the user at a glance: service name, company name, functionality, etc
Post-conditions	N/A
Requirements map	F11, F59, F65
Related Mock up	N/A

Use Case: DD-U08	**Name: Find service dynamic data**
Description	ADVENTURE user wants to see services dynamic data such as ATP, CTP, processes involvement, instances status, etc.
Actors	ADVENTURE user
Pre-conditions	N/A
Inputs	• Service identity
Events sequence	• The user initiates “browse service dynamic details” functionality. • The system connects to the respective third party systems (directly through service gateways or using Monitoring module) • The system retrieves services’ dynamic information from the third party systems. • The system displays the information
Extension points	• Use Case DP-U10 “Find Process Models”,
Outputs	• The information contains dynamic details about the service such as actual ATP or CTP, processes involvement, instances status, etc
Post-conditions	N/A
Requirements map	F11, F59, F65
Related Mock up	N/A

Use Case: DD-U09	Name: Search by QoS criteria
Description	ADVENTURE user wants to find services within ADVENTURE that fit certain Quality of Service criteria (this might be carbon footprint, quality guarantees, reliability, availability, etc).
Actors	ADVENTURE user
Pre-conditions	N/A
Inputs	• QoS Search criteria

Events sequence	• The user specifies QoS search criteria choosing from predefined parameters.
	• The user initiates search.
	• Default: The system retrieves ADVENTURE services that fit the criteria.
	  • The system displays a list of services.
	• There are no services that fit the QoS criteria
	  • The system notifies the user that no appropriate services have been found.

Extension points	N/A
Outputs	• The provided list contains details that are related to the QoS search criteria that orient the user at a glance: carbon footprint, quality guarantees, reliability, etc.
Post-conditions	N/A
Special requirements	N/A
Requirements map	F11, F59, F65
Related Mock up	N/A

Use Case: DD-U10	Name: Find process models
Description	ADVENTURE user wants to find suitable process models within ADVENTURE that fit certain criteria (this might be domain, industry, partners, service, etc).
Actors	ADVENTURE user
Pre-conditions	• There are process models available • The user has permissions to view process models
Inputs	• Search criteria, user identity

Events sequence	• The ADVENTURE user opens a process models search form.
	• The ADVENTURE user specifies search criteria choosing from predefined parameters.
	• The ADVENTURE user initiates search.
	• Default: The system retrieves ADVENTURE processes that fit the criteria
	  • The system displays a list of processes.
	• There are no processes that fit the criteria
	  • The system notifies the actor that no appropriate processes have been found.

Extension points	• Use Case DP-U12 “Browse process details”
Outputs	• The provided list contains generic details that orient the

	user at a glance: process name, industry, etc., and comprise the selected search criteria.
Post-conditions	N/A
Special requirements	N/A
Requirements map	F56, F57
Relevant Main and other Tasks re events	• Main: T4.2; • Other T4.1
Related Mock up	Figure 53 - Search UI

Use Case: DD-U11	**Name: Find process model templates**
Description	ADVENTURE user wants to find suitable process model templates (best practices) within ADVENTURE that fit certain criteria (this might be domain, industry, etc).
Actors	ADVENTURE user
Pre-conditions	• There are process model templates available • The user has permissions to view process model templates
Inputs	• Search criteria, user identity

Events sequence	• The ADVENTURE user opens a process model search form.
	• The ADVENTURE user specifies search criteria choosing from predefined parameters.
	• The ADVENTURE user initiates search.
	• Default: The system retrieves ADVENT|URE process model templates that fit the criteria.
	  • The system displays a list of process model templates.
	• There are no process templates that fit the criteria
	  • The system notifies the actor that no appropriate process templates have been found.

Extension points	• Use Case DP-U12 “Browse process details”
Outputs	• The provided list contains generic details that orient the user at a glance: process template name, industry, etc., and comprise the selected search criteria.
Post-conditions	N/A
Special requirements	N/A
Requirements map	F56, F57
Relevant Main and other Tasks re events	• Main: T4.2; • Other T4.1
Related Mock up	Figure 53 - Search UI

Use Case: DD-U12	**Name: Browse process details**
Description	ADVENTURE user browses the details of a process model that includes the activities and services bound
Actors	ADVENTURE member or Guest
Pre-conditions	• The user is authenticated and authorized for the operation

Inputs	• Process model/Process model template, user identity

Events sequence	• The user invokes browse process details by selecting a process model from a list.
	• The system retrieves process model static data – process model elements and the respective information.
	• The system checks whether the user has sufficient permissions to see the services bound to process model activities.
	• Default: The actor has sufficient permissions
	  • The system retrieves services bound to process model activities.
	  • The system displays a read-only view of the process model and its elements and services in the Process editor.
	• The actor does not have sufficient permissions to see the services bound to process model activities
	  • The system displays only the process model and its elements

Extension points	N/A
Outputs	• A detailed view on service details.
Post-conditions	N/A
Special requirements	N/A
Requirements map	F71, F71A
Relevant Main and other Tasks re events	• Main: T4.2; • Other T4.1
Related Mock up	N/A

5.4.4 Component Interaction

The Data Provisioning and Discovery component provides the necessary high-level data for maintaining smart processes for virtual factories including partners, services, processes data, as well as the corresponding annotations. In order to provide this functionality the component interacts with other ADVENTURE functional components as follows:

- *Dashboard* component realizes the Data Provisioning and Discovery UI allowing the user to effectively define, maintain, publish and discover the ADVENTURE data.
- *Process Designer* component realizes the ADVENTURE functionality for interactive Smart Process definition, publication and maintenance.
- *Cloud Storage* component ensures the persistency, maintainability and accessibility of the ADVENTURE data defined and processed by the Data Provisioning and Discovery component.
- *Process Optimization* component allows the user to construct optimal manufacturing Smart processes according to different criteria.

The diagram below summarizes the dependences between Process Designer and other ADVENTURE functional components.

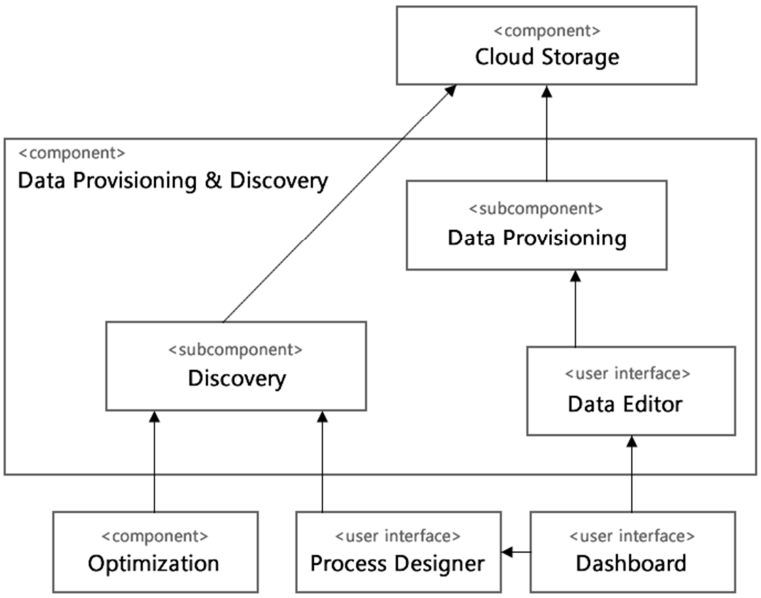

Figure 55 - Process Designer Interactions

5.4.5 Conceptual Data Model

The Data Provisioning and Discovery component defines a suitable data model in order to support efficiently ADVENTURE tools and users in their information management activities.

In brief, the key objects of data modeling are *ADVENTURE members* (as clients, suppliers), the *services* they provide and the *processes* in which they are involved. Each of these is associated further with other first-class citizens in the domain model that help to shape a holistic view to aid the operations of the ADVENTURE system.

- The *ADVENTURE Member* descriptions aggregate a number of different types of information in order to specify organization profiles, suitable for operations within AFVENTURE. For example, they provide sufficient level of detail to enable the ADVENTURE Broker to determine whether a partner is eligible to participate in a specific manufacturing process from e.g. legal, operational, organizational, commercial, geographical point of view.
- The *services* descriptions provide sufficient base to query for compatible services that match process or process step requirements and facilitate the discovery and invocation of these services for higher automation and adaptability at run time. *Services* are provided by *members* and are associated with their profile. *Services* produce *products*, which describe the effect of its invocation and are key search criteria when looking for a service. A product in ADVENTURE is described by means of using external classifications (NACE, e@Class,etc) and helper artifacts (attached specifications, images, etc.). Services are related to the processes in which they are used by the operations they aggregate. Services are described in terms of conditions (to execute), effects (after execution) and different non-functional properties, to aid the automation of matchmaking of services to process goals.
- *Process* descriptions support the processes search, the management of best practice process templates, sharing and automatic recommendations and process adaptation.

Each process model comprises a number of logically related *Modeling Elements* (events, activities, flows,etc.). The manufacturing *services* are bound to the *activity* subtype of *modeling elements.* Activities are described by conditions, effects and non-functional properties, similar to services, in order to perform a reliable matchmaking for the binding.

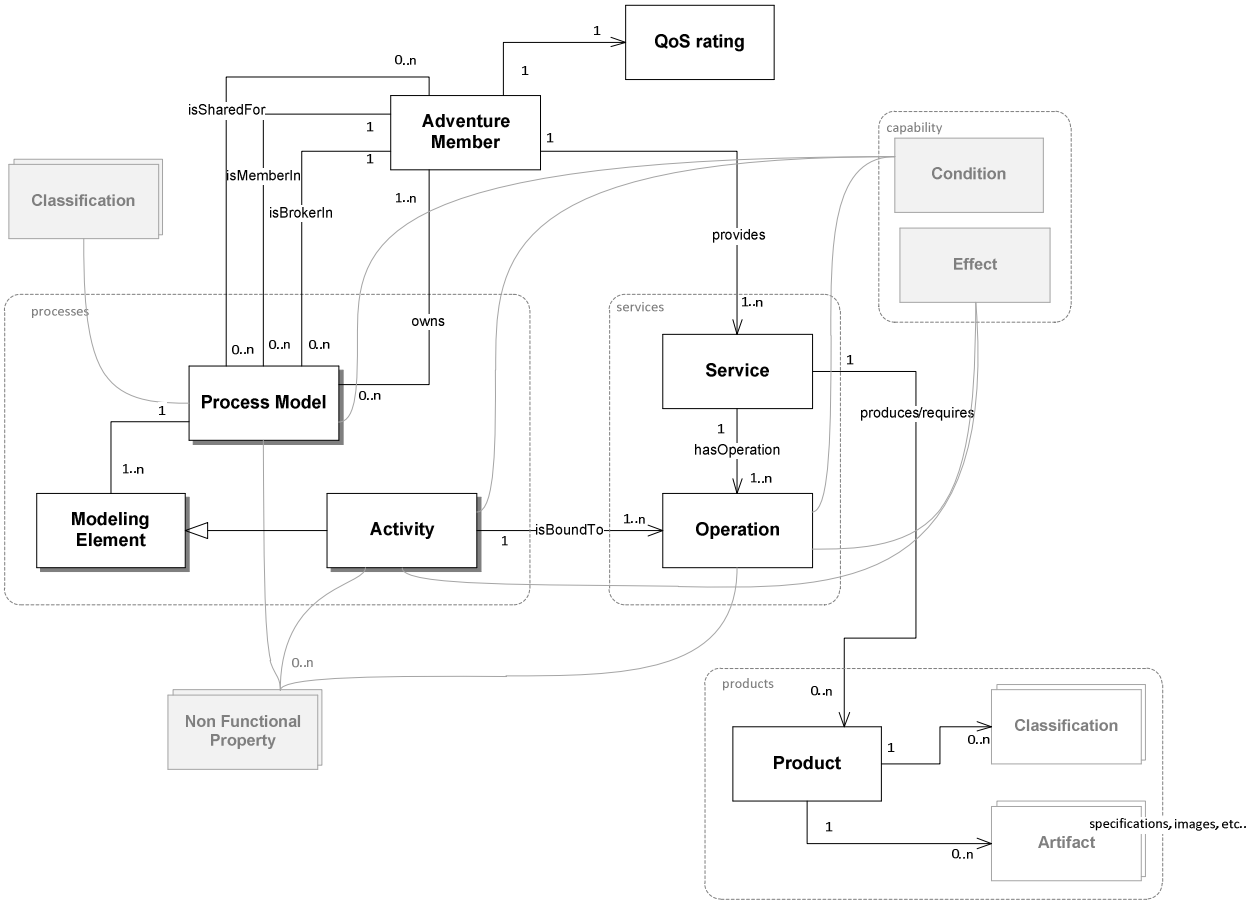

Figure 56 - Process Design Domain Model

5.4.6 Parameters to Take into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	++
Regularly Updated	+
Technical Up-to-Dateness / Appeal	+++
Open Source	+++
Non-Infecting	+++
Code-Quality	+++
Extensibility	+++
Community	+++
Performance	++
Reuse of existing developments	+/-
EU project origin	+/-
Platform (Portability)	+
Open Standards Compliance	+++
Interoperability	+++
Specific parameters	
Mapping to domain-specific data	++
Business Intelligence capabilities (analytics, reporting, data mining	+/-
Standard query language	+++
Support for complex queries	++
Semantic, semi-structured and structured data models.	+++
Non-disruptively extensible data model	++
Annotations support	+++
Reasoning support	+/-
Lightweight service standards compliance	++
Cloud readiness	++
Remote data storage service access	++
Usability	++
LinkedData principles support	+/-

5.5 Message Routing

The Message Routing component is situated in the Data Exchange layer of the system architecture and will manage the secure exchange of messages and files between components and users/components.

5.5.1 Overall Functional Characterization

The Message Routing component will be a server application that manages message buffering, synchronous and asynchronous message delivery, binary file transfer and communication to other Message Routing components. With a final production system these could be in a server cluster of interacting single applications to allow for failovers and elasticity for the messaging.

Each external component instance, such as the Optimizer, will have a role, which is its component type and identifies the functionality of the component. Messages can be sent to such a component role and the Message Routing component will then ensuring the routing the message between the sender and a receiver.

Each component will also have a unique identifier, which will be created by the Message Routing component during the component registration process and enables components to also message other component instances of the same type if necessary and also enables the Message Routing to send a Message based on role instead of ID. For example, if there exist three Cloud Storage components in ADVENTURE, it is possible to send a message to the Cloud Storage component using the role without knowing the single component instances IDs. In this case the Message component will select the Cloud Storage component and send the message. The use of this functionality is that not every single component has to implement it's own logic to check whether a specific component of a certain role is registered and reachable and if not to handle the problems, but instead can leave this to the central Message Routing component. If the message should be delivered to one specific component instead, the unique identifier has to be used.

Each message and message exchange will also have a unique identifier, so that components can keep track of message conversations by ID. The difference between a message and a message exchange needs to be made, as the Message Routing will send a lot of different messages that it needs to identify, while a component only wants to identify which messages belong together. For example, if a component send four different messages, each has it's own ID and each answer has it's own ID, but each individual ID will be different. The message exchange ID of the outgoing message and the response will match though.

Each user will also have a unique ID by which users can directly communicate.

The messaging component is not aware of any message content, except the header information (e.g. sender, recipient, message type etc).

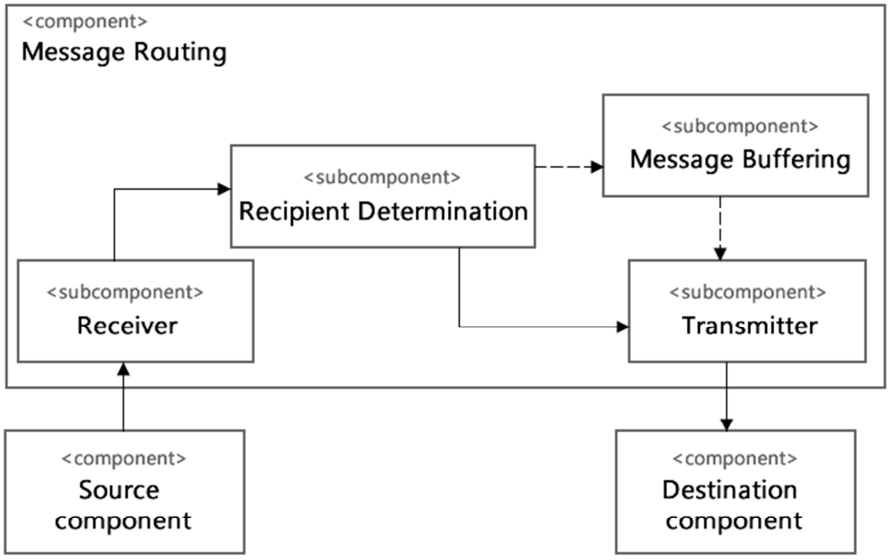

Figure 57 - Architecture Diagram of the Message Routing Component

The Message Routing component needs to contain several functional subcomponents:

- **Source component** is the ADVENTURE component that sends messages and files to the receiver component and that can be invoked by other components.
- **Receiver** is a subcomponent that receives a Message from a source.
- **Recipient Determination** is a subcomponent that determines based on the destination data to which target components and/or users the message needs to be transferred to. The destination data may be inferred from process data.
- **Message Buffering** is a subcomponent that stores a message and associated file for transfer until the message and associated files can be properly transmitted to the destination component.
- **Transmitter** is a subcomponent sending the message to the destination, receiving an acknowledgement of successful transfer and forwards this acknowledgement to the Source component.
- **Destination component** is an ADVENTURE component that receives messages and files, forwarded by the Message Routing and generating an Acknowledgement back.

5.5.2 Use Cases

The following functionalities have been identified from the requirements:

- **Provide secure communication**

  The default communication in ADVENTURE will be provided over a standard secure protocol layer (like SSL).

- **Provide message buffering**

  The Message Routing components saves messages and associated information, if the destination is offline (which can be determined technologically if a technology with presence detection is used or when the transmit call times out) and automatically sends the messages, once the destination is available again (which can be determined technologically if a technology with presence detection is used or via polling).

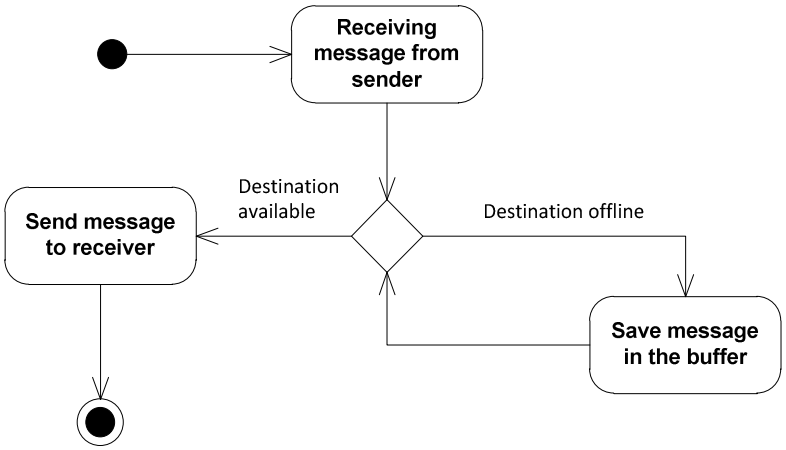

Figure 58 - Activity Diagram Message Routing with Buffering

- **Enable component registration**

  A new component instance (e.g. a new Gateway instance) will be able to register within the Message Routing component.

- **Provide catalog of registered components by role**

  To be able to address the available components, it will be possible to get a list with all components of a specific role.

- **Provide integration for components (e.g. Gateways, etc.)**

  A client library or a Web Service will be provided to other components to make frictionless integration of the Message Routing possible. The integration can be performed by an API call or by a GUI which could be integrated in the dashboard.

- **Deliver messages between components**

  This is the core functionality of the Message Routing component. It provides synchronous and asynchronous communication, in which the communication between the Message Routing component and the destination component is always asynchronous. This can be seen inFigure 59.

- Sending Acknowledgement Messages

  The destination component sends an acknowledgement back to the source component, but it is not guaranteed that the answers will be received in the same order as they were sent. So the source component has to assign the answers to the requests autonomously.

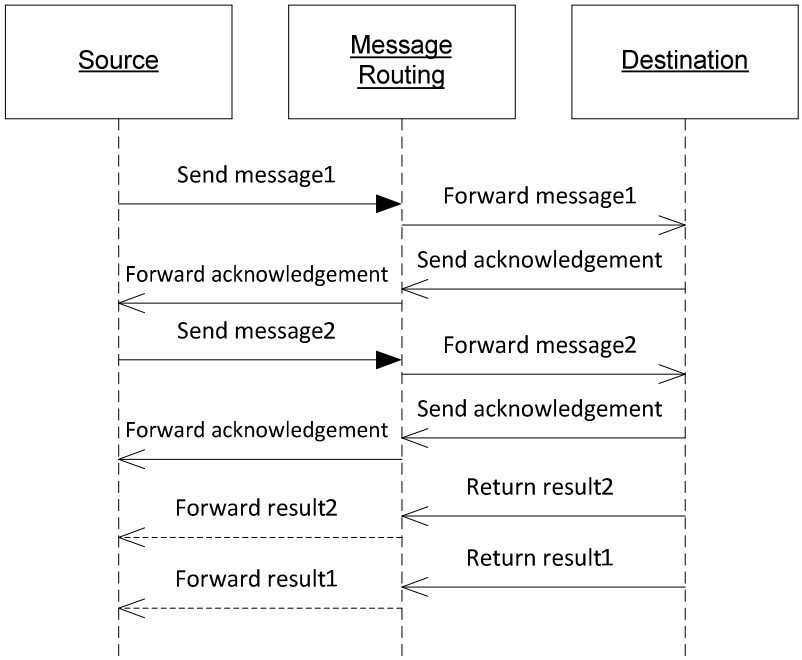

Figure 59 - Sequence of Results and Acknowledgements Unordered

- **Deliver messages from a component to users**

  It will be possible that components can send human-readable messages to users based on filtering criteria, user-id, user association and/or user role. This is needed for alerts users have to receive.

- **Provide Instant Messaging functionality for users**

  Users will be able to send human readable messages to other users. Instant Messaging functionality implies other functionalities like presence-awareness, friend-lists, blocking of users, group-messaging, etc. As this is the only identified functionality with a low priority, not all of these implied functionalities might be addressed. This is addressed in the Message Routing component because this functionality cannot be solved anywhere else.

The fore-mentioned functionalities imply the need for the following user interfaces:

- **Message Routing Configuration View**

  A configuration view for the Message Routing will be provided that lists components and enables administrators to add component instances.

- **Message View**

  A sub-view for receiving human-readable messages will be created.

Other predicted functionality derived from the requirements that may be necessary:

- **Provide User Admin-functions**

  Administrative API and UI for CRUD of users within the Messaging system will be available, or it might be part of the Data Provisioning & Discovery component, where the rest of the user data is set.

- **Provide binary data transfer**

  Binary data will be needed for documents, and might also be needed by the Transformation Services to receive external communication from Gateways. Huge binary files may have to be cached in the buffer till they are transmitted.

- **Provide Multi-Recipient Messaging**

  An API for sending Messages to multiple recipients will be provided when e.g. Process Monitoring informs all users of a Virtual Factory or all users of a company about something e.g. an issue that needs to be handled.

- **Provide Role-based Messaging**

  An API to send Message to a not-specified component of a certain type will be needed – for example when a component wants to save data in Cloud Storage but doesn't want to address a specific instance of Cloud Storage.

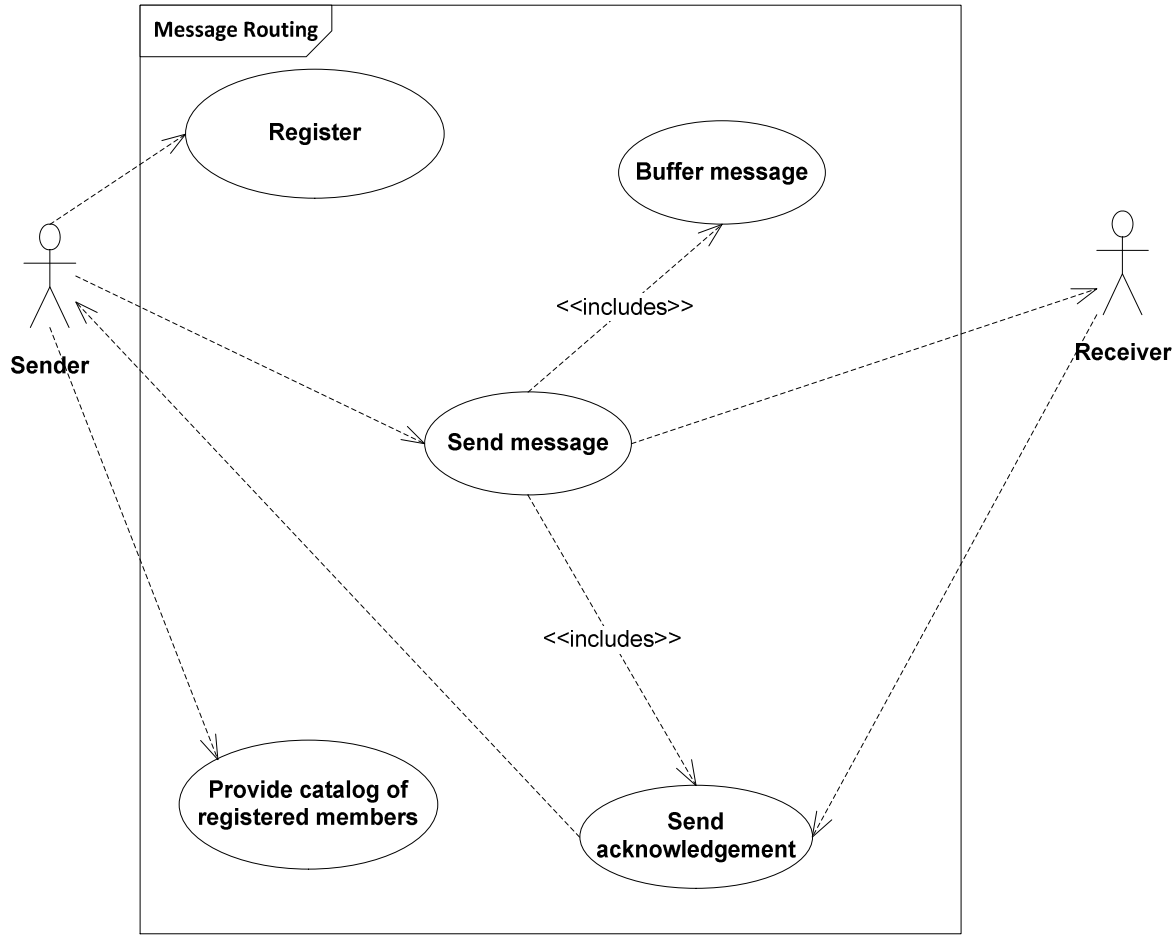

Figure 60 - Message Routing Use Case Diagram

Use Case: MR-U01	**Name: Send message with Buffering**
Description	This is the core functionality of the Message Routing component, sending messages from one actor to another. If messages cannot be delivered because the receiver is offline, they will be saved and sent, if the receiver is available. For every received and routed message an acknowledgement of receipt will be returned to the sender
Actors	ADVENTURE component(s), ADVENTURE user(s)
Pre-conditions	Sender and Receiver are registered in the Message Routing
Inputs	Message to transfer, destination ID or role
Events sequence	• Component transfers message and destination. • Message is transmitted to destination. • Default: destination is available • If destination not available:

	• Data is buffered • A notice is sent to sender • When destination becomes available, go to 2.). • The destination sends an acknowledgement and/or the response data. • The response is transmitted back to the sender. • Default: the original sender is available • If the original sender is not available: • Response is buffered • When original sender becomes available, go to 4.). • An acknowledgement is sent to the sender. • If the original sender is not available, the acknowledgement is buffered → when original sender becomes available, go to 5.).
Extension Points	N/A
Outputs	• Message • Result • Acknowledgement message
Post-conditions	N/A
Special requirements	N/A
Requirements map	F16, F28, F29, F30, F32, F37, F85

Use Case: MR-U02	Name: Register message participant
Description	A register service will be provided to register new participants, which want to be able to exchange messages in ADVENTURE.
Actors	ADVENTURE component
Pre-conditions	N/A
Inputs	Unique ID, Reference to data entity in Cloud Storage, Password
Events sequence	The service for registering a new message participant is called The ID is checked for uniqueness Default: it is unique If it is not unique, an exception is returned, asking for another ID The participant is registered and it's data is included in it's data representation in the Cloud Storage An Acknowledgement is returned
Outputs	Notification message, if the registration was successful
Post-conditions	N/A
Special requirements	N/A
Requirements map	F27

Use Case: MR-U03	Name: Provide List of registered members
Description	A list of subscribers can be requested, which contains all members that are registered at the Message Routing and can receive messages, regardless, if they are online or offline. This is necessary if a component wants to find out if there are components of a certain role registered, or if a certain Gateway is registered
Actors	ADVENTURE component
Pre-conditions	N/A
Inputs	N/A
Events sequence	• List is requested • List is returned
Outputs	List with registered message participants, including their role
Post-conditions	N/A
Special requirements	N/A
Requirements map	F27

5.5.3 Component Interaction

The Message Routing component interacts with all components directly by forwarding messages. The calls that it needs to handle are shown in Figure 61.

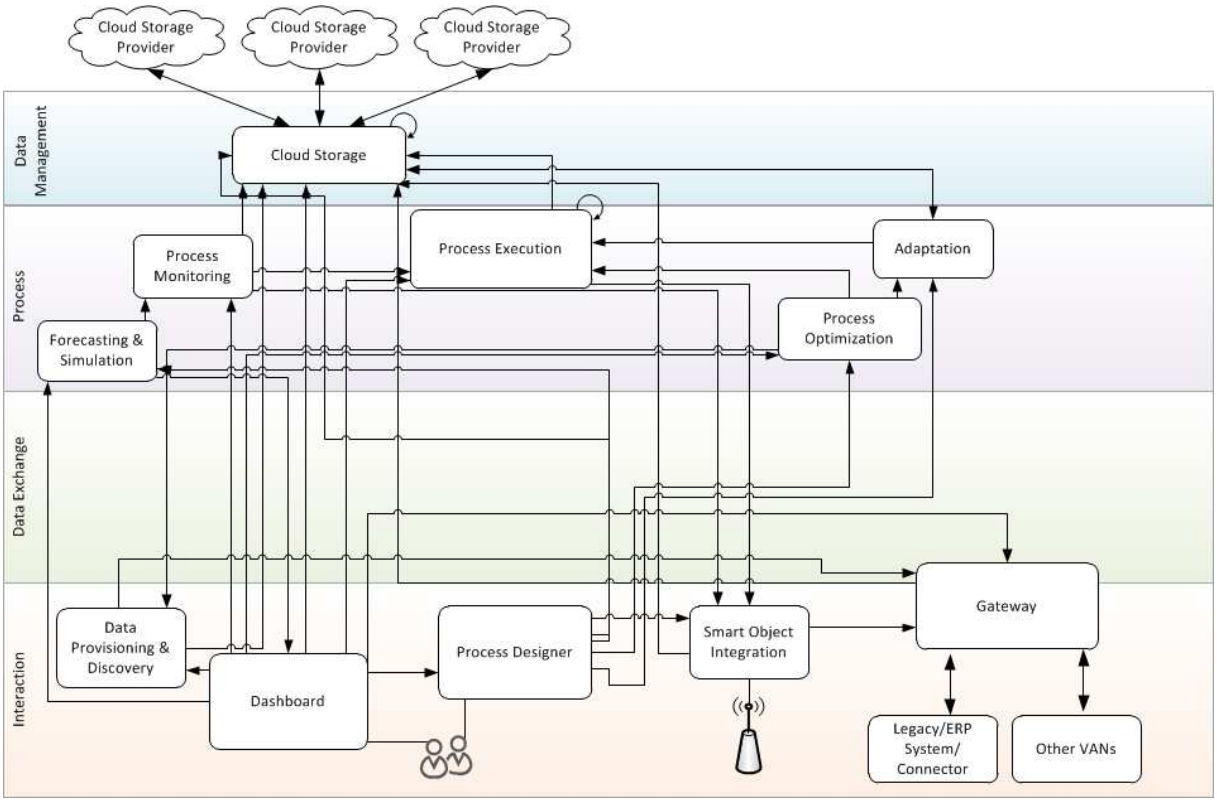

Figure 61 - Logical calls of components inside ADVENTURE

An API or a client library that can easily be used by all components is necessary to guarantee security, acknowledgement and correct use of Message Routing. This needs to be in place for communication as requested by the requirements.

5.5.4 Conceptual Data Model

The main data format for the Message Routing, the Gateways and the Transformation Services components is the format of the data transferred by the messages.

The data format will be structured, reusing an existing protocol specification if possible. Messages will contain their message ID, a message exchange ID for replies, a sender ID, a destination ID, a sent-when and an expires-by date-time-stamp, to power various features that the Message Routing is expected to deliver. It will also contain a main message payload element, which can contain more structured or unstructured message information.

Other components data models will need to adhere to this envelope model when designing their communication strategy and components will need to provide the necessary information to populate it.

5.5.5 Parameters to Take Into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+++
Regularly Updated	+
Technical Up-to-Dateness / Appeal	++
Open Source	+
Non-Infecting	++
Code-Quality	+
Extensibility	+++
Community	+
Performance / Scaleability	++
Reuse of existing developments	++
EU project origin	--
Platform (Portability)	+++
Open Standards Compliance	+
Interoperability (easy intergration for all platforms)	+++
Specific Parameters	
Reliable Messaging / Receipt Acknowledgement	+++
Provide binary message exchange	++
Secure communication protocol	+++
Signed messages	+/-
Provide message buffering	+++
Point-to-Point Messaging for Component Instances	+++
Ability to treat users as message partners	++
Multi-recipient-messaging	++
Multi-instance-Server for cloud hosting	++
Provide configuration UI for routing	+
Provide user-message UI	-
Message Partner Presence Awareness	+
Uses UFT-8	+++
Lightweight Infrastructure for Remote SME Endpoints	++
Communication Partner Registry included	+

5.6 Transformation Services

The Transformation Service in the ADVENTURE Platform is responsible for transforming information between different data content formats. It runs in the data exchange layer, linked to the message routing system.

5.6.1 Overall Functional Characterization

The main functionality of this component is to transform external data formats, typically message serial formats like XML, sent through the ADVENTURE Gateways (e.g. by ERPs), to be used by internal ADVENTURE components. The Transformation services will typically be invoked by the Process Engine but can also be invoked by other components within ADVENTURE where they have a need for data transformation. The descriptions of the available transformations are published in the Cloud Storage component so that the different modules can know about them (e.g. can be used at the process designer).

Transformation is a feature with a specific vocabulary as follows:

- Transformation is the total actions necessary to convert syntax or content A(s) into syntax or content B(s) (the 'S' is used since there can be multiple inputs and outputs).
- Mapping is the design time act of creating the commands/instructions/code to make this conversion. Typically a mapping file will contain the source scheme, output schema and the linkage between them both. The mapping file format is specific to the Translation engine.
- Translation is the run time act of morphing instance data from As to Bs according to the Mapping.
- The Translation Engine is the software engine that performs the translation; typically this may be either:
    - A generic Translation engine, that can run maps related to that specific engine at runtime (or an index) to the map as well as input files and output locations
    - A specific Translation engine which can run, or is created, to run just that map at runtime and is provided with input files and output locations

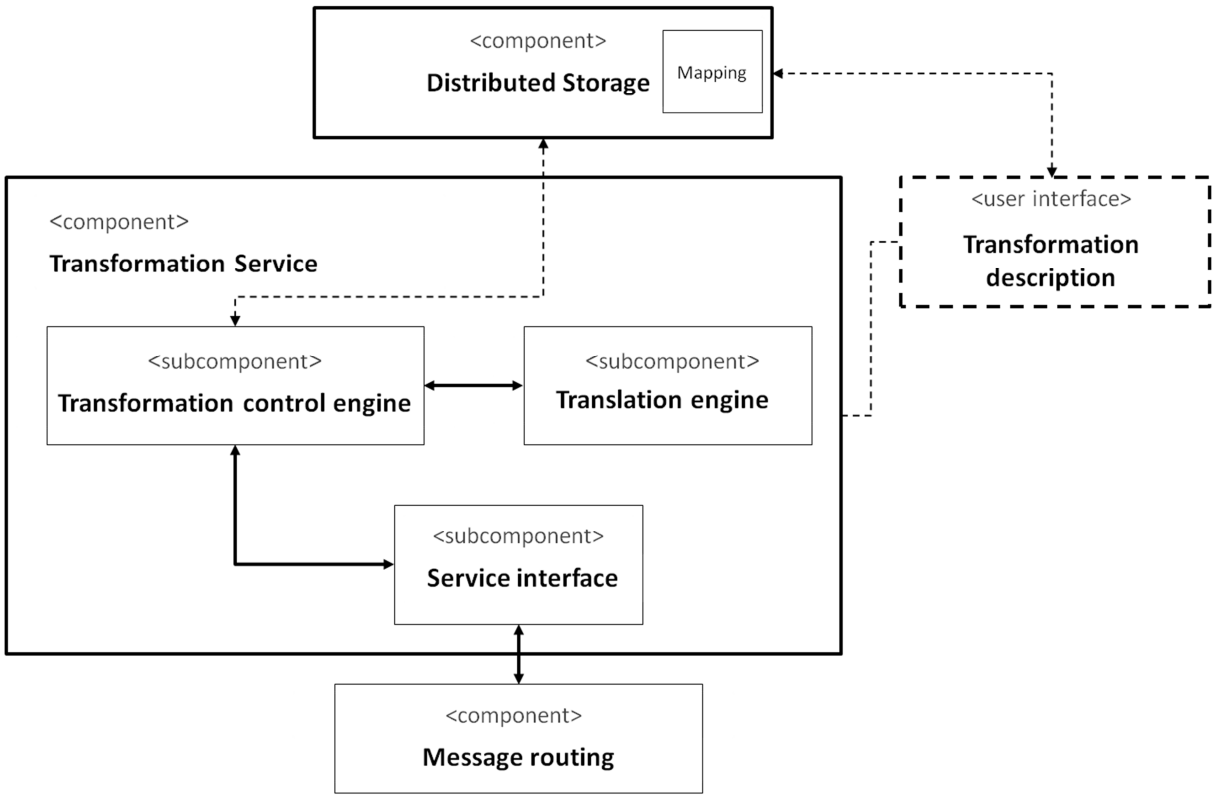

Figure 62 - Transformation Service Component Diagram.

The Transformation Service is based on the following subcomponents and data:

- Translation engine: The translation engine is the main component of this functionality. Based on entry information in format A, and knowing the pre-created map on which the transformation is based, it is able to transform it to information in format B, by executing the transformation instructions (Mapping). Note that it is not in the scope of ADVENTURE to create this, and an existing engine of Partner TIE will be customised to perform this. Further engines could also be plugged-in to the architecture.
- Transformation control engine: Consists on auxiliary functionalities that allow recovery and checking of the necessary information (source information, check format of source information, recover mapping file from format A to format B from the storage system, etc...) so that the translation engine could be executed successfully.
- Service interface: Is the component that exposes, and that allow this component to communicate with other ones, and to be invoked (e.g. ADVENTURE components such as Execution engine).
- Mapping (data): Is a syntax that contains the specific actions to be followed by the transformation engine in order to transform information in format A into information in format B. Mapping files are edited off line (out of ADVENTURE Scope) by mapping experts. These mapping files must be compatible with the Transformation engines which are used in relevant ADVENTURE processes (parts) they participate in. The mapping files will be stored at the ADVENTURE distributed storage system.

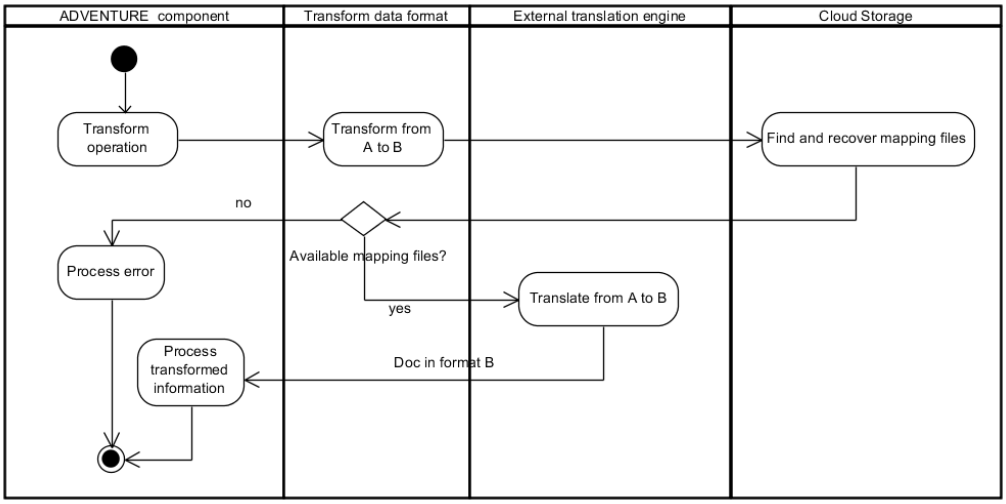

Figure 63 - Transformation Services Activity Diagram

5.6.2 Use Cases

Starting from the requirements document, the "Provide data transformation" functionality has been identified and will make use of the following user interfaces:

- A mapping editor where one information format is mapped to a second information format (as this part of the functionality relies on existing commercial products, it will be considered as a black box and will not be specified here).
- A transformation services description view, which identifies how the transformation service can be reached, the different data format transformations it supports, the additional files needed to perform the transformation actions (mapping files),etc...

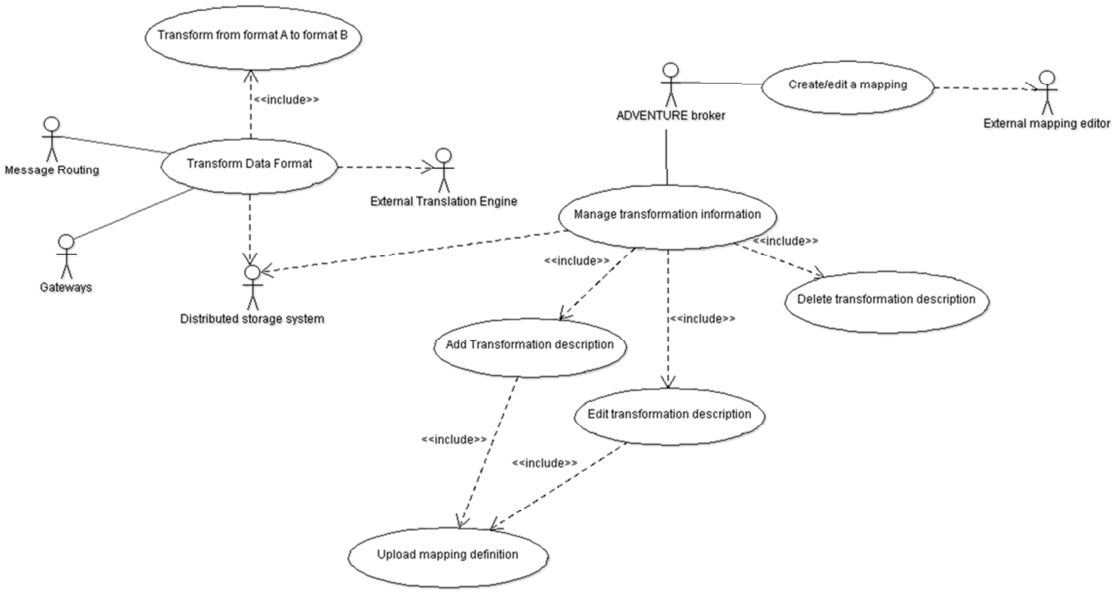

Figure 64 - Transformation Service Use Cases.

In the following tables, the main use cases for the Transformation Service are described.

Use Case: TS-U01	**Name: Create/Edit a mapping file**
Description	As a prerequisite, a transformation engine needs to know about how to perform transformations between different data formats. The transformation instructions are usually expressed by means of a mapping file. This functionality allows a technical user to define or modify a mapping file. Note: Whilst this needs to be performed, it is outside the scope of ADVENTURE.
Actors	ADVENTURE broker
Pre-conditions	• Availability of the input and output schema formats
Inputs	• Input and output schema formats • Knowledge of relationships between input and output formats.
Events sequence	• Broker creates or edits a mapping file using an external mapping editor. • The Mapping file is saved for further usage in the ADVENTURE platform.
Extension points	N/A
Outputs	• New / Updated mapping file.
Post-conditions	N/A
Requirements map	F8,F18,F19
Related Mock up	N/A

Use Case: TS-U02	**Name: Transform data format**
Description	Is a black box functional element that provides the core functionality of transforming a document in format A(s) to a document in format B(s) by means of some intermediate data (mapping file) and through an external translation engine. The Transform data format is a wrapper for the specific "Transform from format A to format B" functionalities stated before (TS-U03).
Actors	Process Execution Engine, External translation engine
Pre-conditions	• The mapping files have to be available at the system
Inputs	• File to transform • Mapping file(s)
Events sequence	• See TS-UCU03.
Extension points	N/A
Outputs	• New / Updated mapping file • Exception / Error message
Post-conditions	N/A
Requirements map	F8,F18,F19
Related Mock up	N/A

Use Case: TS-U03	Name: Transform from format A to format B
Description	Based on an input file (or message), a mapping file identifier (or mapping file) and an external translation engine, it transforms the input formats(s) into the desired format(s) of the output format, and returns it to the caller.
Actors	Process Execution Engine, External translation engine.
Pre-conditions	• The mapping file for transforming from A to B is accessible at the system.
Inputs	• File to transform • Mapping file(s)

Events sequence	• Triggering of the launch of the Transformation service (though the message routing)
	  • Default: Process Execution Engine
	  • Other components
	• The mapping file and auxiliary data needed is found and fetched from the storage system.
	• Transformation engine service uses the mapping file to transform from A to B by means of the external translation engine.
	• The input transformed into B is delivered to the caller system via Message routing and as directed by the Process Designer.

Extension points	N/A
Outputs	• New / Updated mapping file • Exception / Error message
Post-conditions	N/A
Requirements map	F8,F18,F19
Related Mock up	N/A

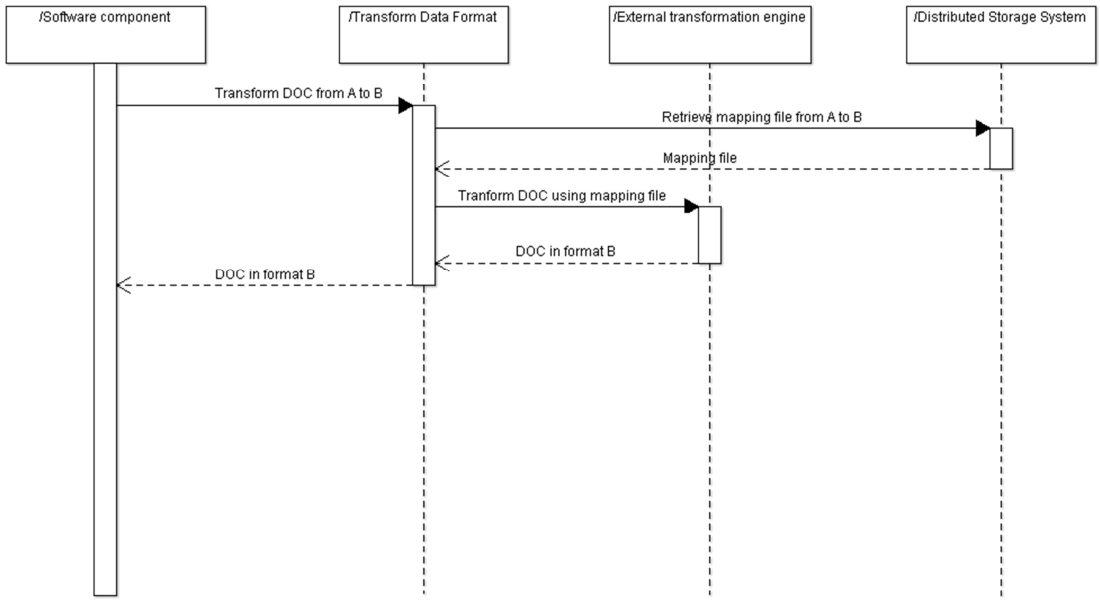

Figure 65 - Transform Data - Sequence Diagram

Use Case: TS-U04	**Name: Manage transformation information**
Description	Functionality that allows a technical user to describe the different transformations available at the ADVENTURE system, so that other ADVENTURE components could make use of them. The user interface to manage the transformation descriptions is provided by the Transformation services component, and exposed via the company configuration interface at the Dashboard component (see use case D-U06).
Actors	ADVENTURE broker.
Pre-conditions	• ADVENTURE User has to be authenticated and have rights to change the mapping information description (meta information).
Inputs	• Data to be added, edited or deleted about a transformation operation, Mapping file(s) to be uploaded.
Events sequence	• Broker reaches the services management section for its organization in the dashboard (see use case D-U06). • They adds/edits/deletes information about a specific data transformation. • Broker uploads a mapping file (see TS-U05) associated to the transformation information, so that this file can be used at the transformation process in runtime. • If it is needed more than one file to perform the specific transformation, the user repeats the action to upload the needed files. • Broker confirms the operation.
Extension points	N/A

Outputs	• New/Updated/Deleted transformation information. • Error.
Post-conditions	N/A
Requirements map	F8,F18,F19
Related Mock up	Figure 66 - Manage Transformation Information Mockup.

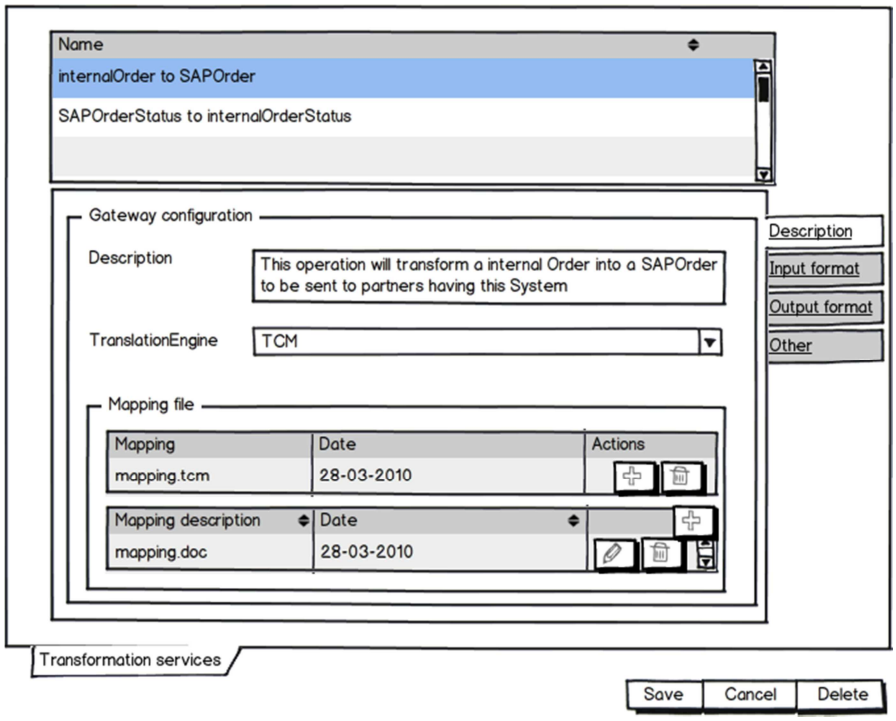

Figure 66 - Manage Transformation Information Mockup.

Use Case: TS-U05	**Name: Upload mapping definition**
Description	When an ADVENTURE broker is adding or editing the information about a specific transformation operation, they have to be able to upload mapping files that has been previously created or edited.
Actors	ADVENTURE broker.
Pre-conditions	• Mapping file runnable by the transformation service is available (obtained at TS-U01). • Broker has to be identified and have rights to update the service files.
Inputs	• User credentials. • Mapping file(s).
Events sequence	• Broker at TS-UC05 performing an adding or editing action. • They are authenticated in the ADVENTURE platform • They reach the services management section for its organisation in the dashboard. • Broker selects the appropriate transformation service. • It chooses the mapping file to be uploaded.

	• Broker uploads the mapping file. • The Broker confirms the operation and the new mapping file is associated to a specific transformation.
Extension points	N/A
Outputs	• Success message. • Error.
Post-conditions	N/A
Requirements map	F8,F18,F19
Related Mock up	N/A

5.6.3 Component Interaction

The transformation engine interacts with the following functional modules:

- The Message Routing system: All the interactions with the rest of the Modules at ADVENTURE Platform will be performed via the Message Routing. So if a specific data transformation is needed e.g by the Process Execution engine (based on user defined Process Model) the transformation request will be launched through it.
- Dashboard: The dashboard component, through the company configuration functionalities will expose the user interfaces provided by this component (see use case D-U06).
- Distributed Storage: The descriptions of the transformation services and associated attachments will be stored using the Distributed Storage component.

5.6.4 Conceptual Data Model

In terms of the data structures, it is necessary to have both a description of the source formats and of the destination formats in order to be able to compose a mapping file.

For the transformation description, it is envisaged the following data types:

- Name: Name of the transformation operation.
- Description: Information about the features of the transformation operation.
- Translator identifier: A Transformation engine can have more than one internal translator engine. This parameter points to the specific one that has to be used for the specific transformation.
- Data source information: Usually this information will identify the data source format, provide technical information about it, and (if necessary) will include a file (or reference to file) containing the technical schema of the data source file
- Data destination information: Usually this information will identify the destination data format, provide technical information about it, and (if necessary) will include a file (or a reference to a file) containing the technical schema of the destination file
- Mapping file information: This will identify the mapping file to be followed by the transformation engine to perform the data transformation. The mapping file information

will provide technical information about it, and will include a reference to the mapping file itself.

- Additional files information: Information about files containing additional information to be used by the transformation engine to perform the transformation process.

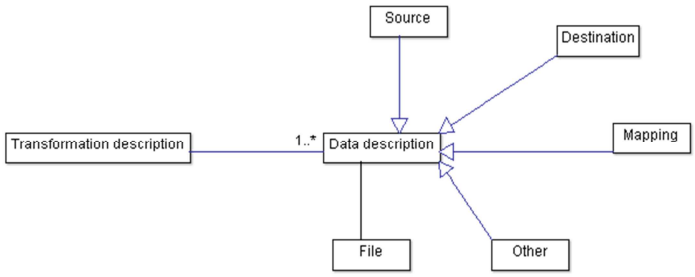

Figure 67 - Conceptual Data Model for Transformation Description.

5.6.5 Parameters to Take Into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+++
Regularly Updated	+
Technical Up-to-Datedness / Appeal	+/-
Open Source	+
Non-Infecting	+
Code-Quality	+
Extensibility	+++
Community	++
Performance	+
Reuse of existing developments	++
EU project origin	+/-
Platform (Portability)	+/-
Open Standards Compliance	+
Interoperability	++
Specific Parameters	
Transformation based on standard	+/-
XML supported	+++
CSV supported	++
EDI like supported	+/-
Database supported	++
Allow semantic annotation of input/output formats.	+/-

5.7 Gateways

ADVENTURE Gateways functionality will provide a framework for building custom made connections to external systems. These systems could either be an ERP system, other legacy systems, Smart Object, etc. The gateway will provide means of communicating with these external systems to allow the exchange of information with the ADVENTURE platform.

5.7.1 Overall Functional characterization

A gateway will comprise of standard components and custom components with functionality developed or created for connecting to a specific external system type and/or instance. A gateways mission is to communicate with a specific system, meaning that a significant part of a gateway implementation is tailored for specific (e.g. SAP ERP) technology or communication/interface protocol.

Gateways are only about connecting 3rd party systems, having an agnostic view about what is the content exchanged and what the format is. Checking and processing of the information gathered through the gateways will be performed at the destination components (as per example the Process Execution component). Gateways will take no decisions themselves. However, gateways can call Transformation Service directly to translate from the external format to predefined (common) ADVENTURE formats which are prespecified.

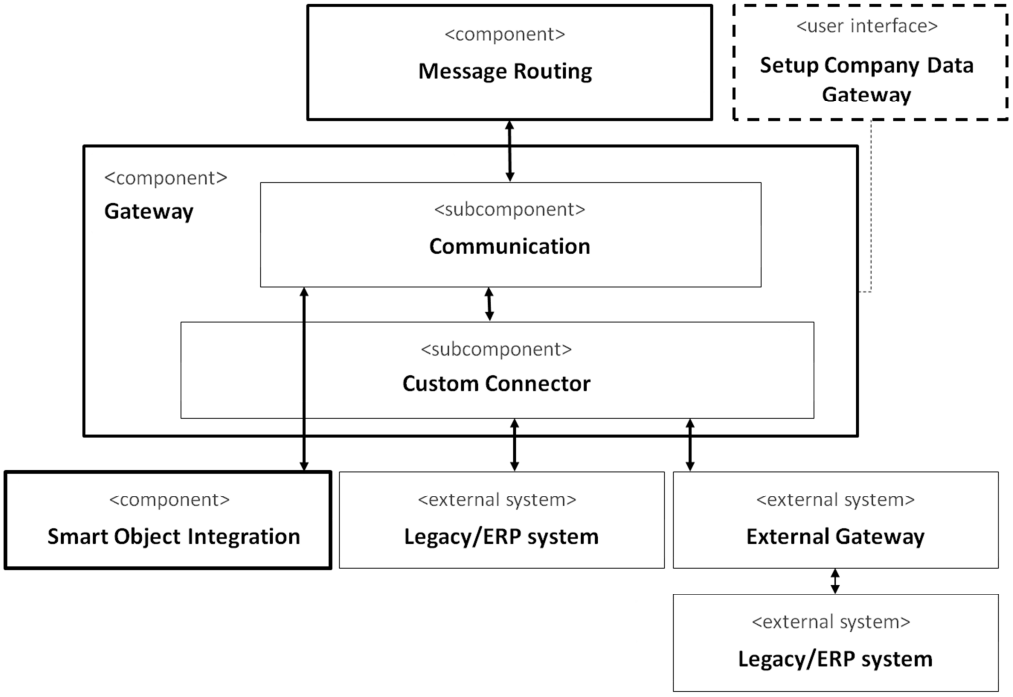

Figure 68 - Gateways General Components Diagram

A gateway is based on the following subcomponents:

- Communication: This sub-component is the bridge to the Message Routing component and will send and receive the messages on behalf of the Gateway. It will use a common

interface to communicate with the Custom Connector component of the ADVENTURE Gateway..

- Custom Connector: Set of customised adaptations that have to be performed in the 3rd Party System side to assure that the data exchanges are performed successfully.
- Smart Object Integration: ADVENTURE component that allow Smart Objects (sensors, etc.) to be integrated into the ADVENTURE platform. It can communicate directly to the Communication subcomponent.
- Legacy/ERP systems: Companies own systems (can also be SaaS systems), that manage company data that should be exposed to ADVENTURE to assure the functionality envisaged at the project. The implementation of these systems are out of the scope of ADVENTURE.
- External Gateways: Already existing gateways that can be used as an indirect way of accessing the companies own systems. The implementation of external gateways are out of the scope of ADVENTURE.

The following user interfaces can be identified linked to the gateways functionality:

- A Company Gateway registration view where companies can register and configure (address, types of messages, etc) their specific gateways instances and services.
-

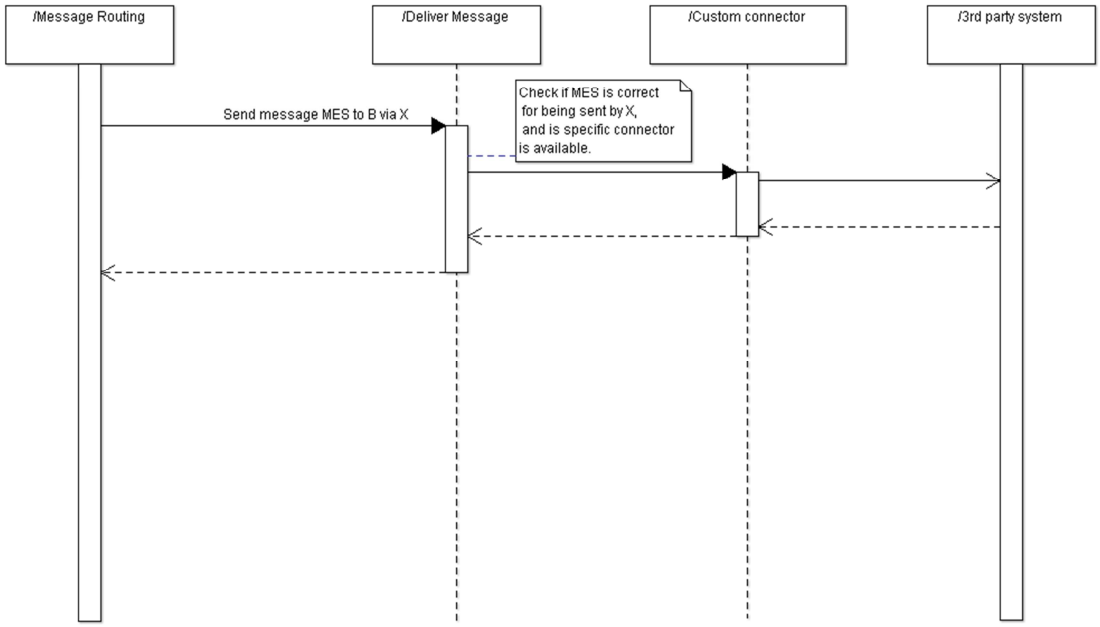

Figure 69 - Gateways General Sequence Diagram

5.7.2 Use Cases

The following functionalities have been identified from the requirements document (D2.3):

- Provide gateway for specific external system
- Provide registration as description functionality for gateway
- Send messages to ADVENTURE network
- Invoke transformation service

These functionalities are depicted in the following Use Case diagram.

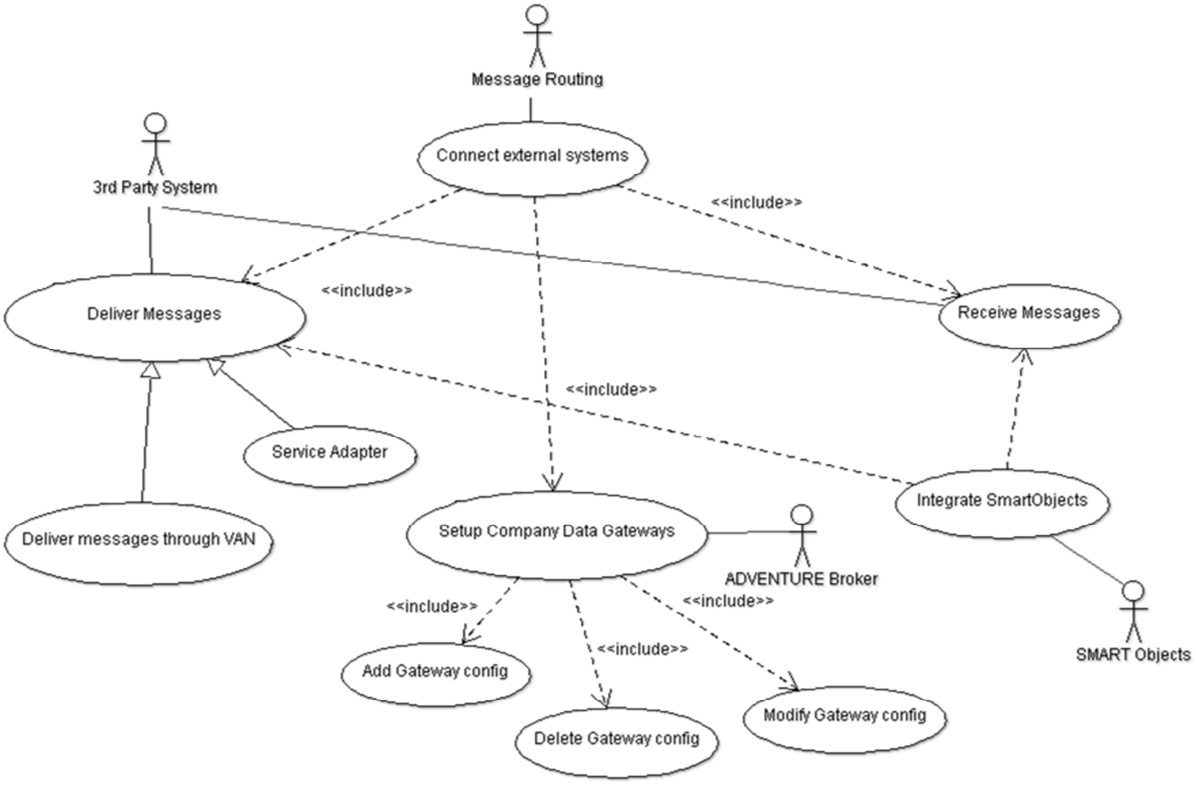

Figure 70 - Gateways Use Case Diagram.

In the following tables, the main use cases for the Gateways are described.

Use Case: GW-U01	Name: Connect external systems
Description	Generic functionality that allows the ADVENTURE platform to exchange information with 3rd party systems and Smart Objects. This generic functionality will allow Message Delivery and Message Receipts.
Actors	3rd party systems, Smart Objects, ADVENTURE Message

	Routing
Pre-conditions	• The company gateway has to be defined in the company profile, and expose specific services.
Inputs	• Messages.
Events sequence	• Described in the specialized Use Cases (GW-U02 and GW-U03).
Extension points	N/A
Outputs	• Message delivered/received successfully. • Acknowledgement. • Exception / Error message.
Post-conditions	N/A
Requirements map	F27, F28, F29, F30, F32, F61
Related Mock up	N/A

Use Case: GW-U02	**Name: Deliver Messages**
Description	Allows the ADVENTURE framework to call services or deliver messages, through specific protocols, (e.g. VAN networks, email, Web Services, etc...) to 3rd party systems. Deliver Messages supports synchronous and asynchronous communication schemes.
Actors	3rd party systems, Smart Objects, ADVENTURE Message Routing
Pre-conditions	• The Gateway service has to be appropriately defined and configured in the company profile (see GW-U04). • Depending on the specific delivery protocol of the gateway, there can be different requirements (e.g. in terms of data in the message, or both actors have to be registered in the VAN, etc...)
Inputs	• ADVENTURE internal message.
Events sequence	• ADVENTURE component wants to make use of a gateway to retrieve/send data from/to an external system. • Through the ADVENTURE message routing, it delivers a message to the appropriate gateway. • The message/call is sent to the destination. • The response from the called gateway is relayed to the message routing, and it finally delivers it to the caller component.
Extension points	N/A
Outputs	• Message delivery ok • Call response • Exception (e.g. time-out)
Post-conditions	• If the specific gateway protocol defines any kind of confirmation from the destination system, it will be processed by the Message Delivery functionality and returned to the internal messaging component for further processing.
Requirements map	F27, F28, F29, F30, F32, F61

Related Mock up	N/A

Use Case: GW-U03	**Name: Receive Messages**
Description	Allow third party systems to submit information into the ADVENTURE framework through a specified (general) communication endpoint which in essence allows communication protocol and enveloping conversion Message submission will be also used by Smart Objects to deliver information to ADVENTURE.
Actors	3rd party systems, Smart Objects, ADVENTURE Message Routing
Pre-conditions	• The Gateway service has to be appropriately defined and configured in the company profile (see GW-U05). • The 3rd. party system has been programmed so that it will trigger a message (or a call) to an ADVENTURE gateway when certain events happen.
Inputs	• 3rd party message
Events sequence	• An event in the 3rd party or Smart Object system happens. • The 3rd party system calls the appropriate ADVENTURE gateway endpoint. • The message is transported by the Messaging Routing to the Process Execution environment for further processing.
Extension points	N/A
Outputs	• Reception OK • Exception / Error
Post-conditions	N/A
Requirements map	F27, F28, F29, F30, F32, F61
Related Mock up	N/A

Use Case: GW-U04	**Name: Setup data gateway**
Description	Different ADVENTURE brokers can configure their companies specific gateways implementations by providing operational information needed to reach and retrieve information from them. For the sake of forecasting and simulation purposes (Forecasting and Simulation Components), an "alternative" service can be associated (not mandatory) to a gateway service main definition. The alternative service will be explicitly used when the gateway service is called from these components (SF-U08). These alternative services will be defined only if the company has implemented them (not further implications for the gateways functionality).
Actors	ADVENTURE Broker
Pre-conditions	• The ADVENTURE Broker is authenticated via the ADVENTURE dashboard and has privileges to manage the company profile information.

	• The company gateway is accessible so that it can be used after the setup process.
Inputs	• Company gateway information (access URL, service description,...).
Events sequence	• The Broker uses the company configuration functionalities of the dashboard and selects the gateway configuration option. • Broker creates/edits/deletes data in specific company gateway configuration. • They upload the necessary files to have a complete description of the service. • Broker saves the changes. The gateway is registered at ADVENTURE platform.
Extension points	N/A
Outputs	• OK • Error
Post-conditions	N/A
Requirements map	F27, F28, F29, F30, F32, F61
Related Mock up	Figure 71 - Basic Gateway Configuration Mockup.

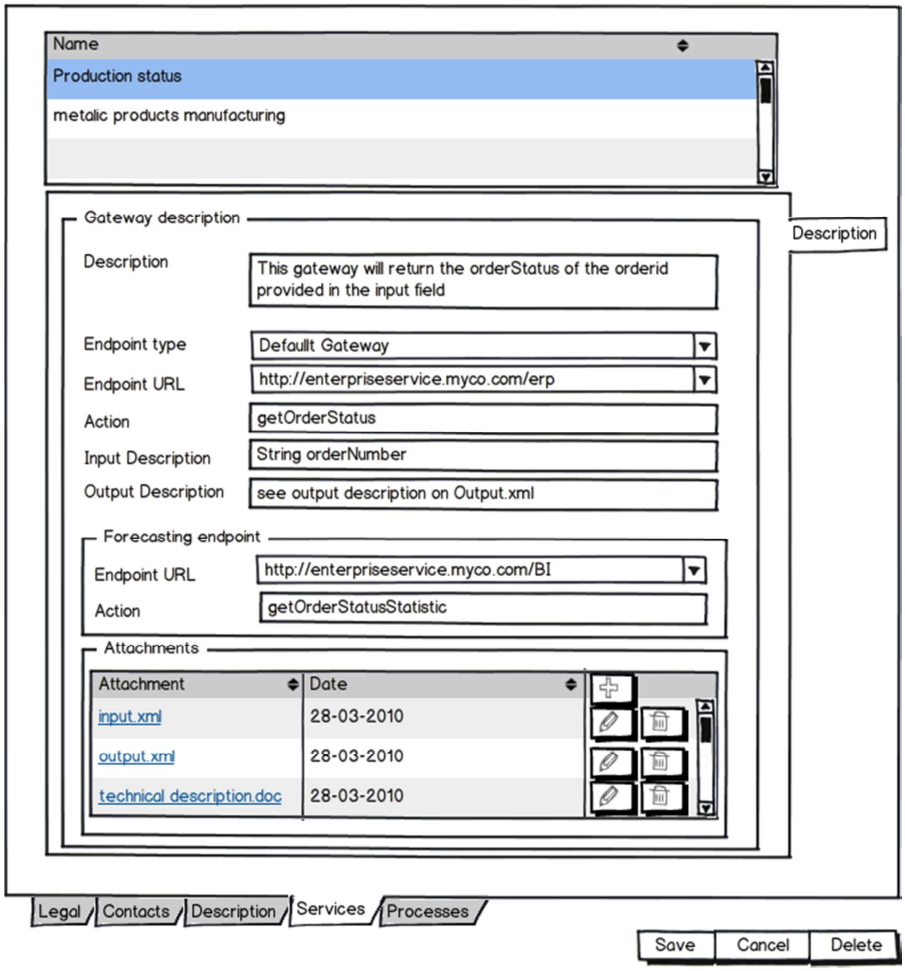

Figure 71 - Basic Gateway Configuration Mockup.

5.7.3 Component Interaction

Main interrelations of gateway elements are related to:

- Dashboard: The dashboard component, through the company configuration functionalities will expose the user interfaces provided by this component (see use case D-U06).
- Message Routing: Gateways interact with the message routing system as a way to receive and deliver the requested information from/to the appropriate ADVENTURE module, several components access the gateways in this way (Process Execution, Forecasting and Simulation,..).
- Smart Objects Integration: Gateways interact with the Smart Objects Integration component so that the information provided by the Smart Objects can be delivered to ADVENTURE components, and also allow these components to query the Smart Objects Integration component for specific data.

- Distributed Storage: The descriptions of the gateway services and associated attachments will be stored using the Distributed Storage component.

5.7.4 Conceptual Data Model

A preliminary description of a gateway service data model can be based on:

- Description: The description of the gateway service, containing narrative information needed to understand what it does, how, what kind of information it delivers, and which company systems are attached to this gateway.
- Input parameters: Technical description of the different parameters that the gateway needs as inputs.
- Output parameters: Technical description of the outputs that the gateway will deliver.
- Linked services: Services linked to the gateway. They can be associated with specific purpose or scope (e.g. substitute the main service when called from forecasting and simulation modules).
- Endpoint information: Information about the endpoint to be used when accessing the custom connector of the gateway (address, type, etc..).
- Attachments: Files containing further descriptive and technical information (e.g. xml schemas of the outputs of the gateway).
- Classification: Classification of the service, to make it easy to find by using services search features.

The technical characterisation of the gateways is intended to be based on, or extracted from a simplification of previous/other research projects or standardisation initiatives.

5.7.5 Parameters to Take Into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	++
Regularly Updated	+
Technical Up-to-Dateness / Appeal	+/-
Open Source	+
Non-Infecting	+
Code-Quality	++
Extensibility	+
Community	+
Performance	+/-
Reuse of existing developments	++
EU project origin	+
Platform (Portability)	++
Open Standards Compliance	++
Interoperability	+
Specific Parameters	
Gateway description based on standard	+/-
Allow semantic annotation	+/-
REST support	+
WS support	++
SMTP support	+/-
VAN support	+/-
Authentication	+/-

5.8 Smart Process Execution

The Smart Process Execution (SPE) component will be at the heart of ADVENTURE, as it will orchestrate all interaction in a virtual factory. Its purpose is to execute Smart Processes, modelled in the Process Designer.

5.8.1 Overall Functional Characterization

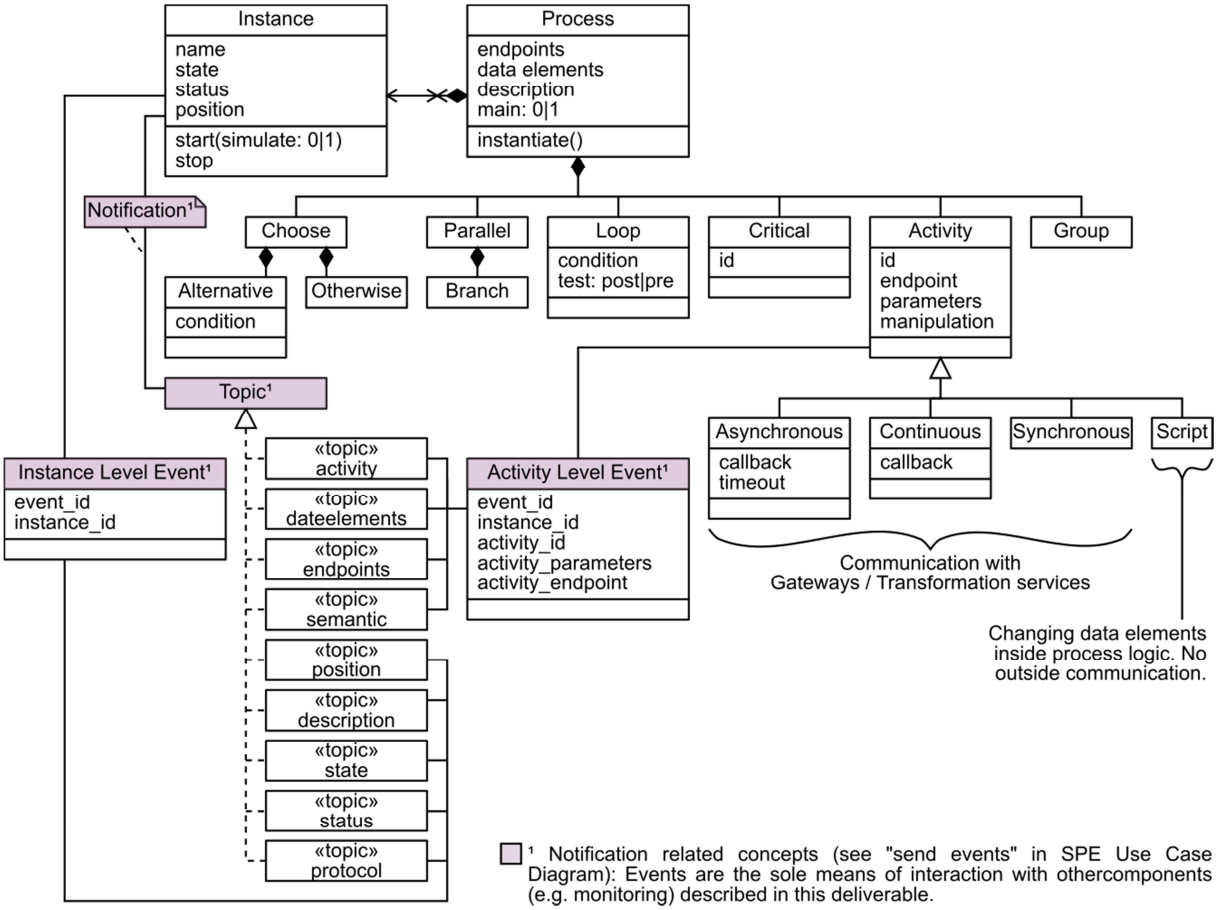

Figure 72 - SPE Concepts and their Dependencies

As shown in Figure 72 the SPE component will deal with Processes, Process Instances and the communication with gateways and logging. From a high level perspective *Processes* and *Instances* have the following minimum requirements:

- Processes*:* Each process has to have a set of attributes, independent of the language in order to capture the process model: Endpoints to enumerate a list of gateways/partners used in a particular process; Data elements (variables) to act as intermediate buffer for capturing results from calls to gateways and providing input for calls to gateways. Description to hold process mode

- Instances: Each process has to be instantiate()'d, each instance has to be stop()-ped and start()-ed. The Name of an instance is intended to hold a description of the circumstances under which the process has been instantiated. The purpose of position is to show a set of activities that are currently executed by the instance. There can be multiple positions because there may be parallel execution of activities. While State will provide a fixed model consisting of a set of conditions and their transition, Status will hold a machine readable semantic/domain specific description of what is currently going on in the process: e.g. "Process is finished, but the results mixed", "Process has an error which is related to machine X", "The results of the process can be found in data element X".

Most important for ADVENTURE will be the differentiation between differentiation between the Standard (main: 0) and Main (main: 1) processes.

- Main processes will be processes that can be instantiated only once (singletons). They will act as main loops for the virtual factory. E.g., when ABB gets a new order, it will enter this order into its ERP system, which in turn will result in an event that will be captured by the running *Main* process via the gateway. These Main processes will then instantiate separate sub-processes to handle the processing of the order.
- Standard processes are processes that will be either
  - instantiated by a *Main* process or
  - instantiated on demand by an ADVENTURE Broker.
- *Standard* processes are instantiable multiple times. For example a standard order process may be instantiated by a Main process (which is connected to an ERP system) when the Main process receives a message about a new order that appeared in the ERP system.

The following sections are dedicated to explain core concepts of the SPE.

5.8.1.1 States

As stated above one of the core functions is the communication of the state of a process instance to the logging component and thus to the ADVENTURE broker. The SPE will provide two different kinds of states:

- Global instance state as show in Figure 73.
- Per activity state, as shown in Figure 74.

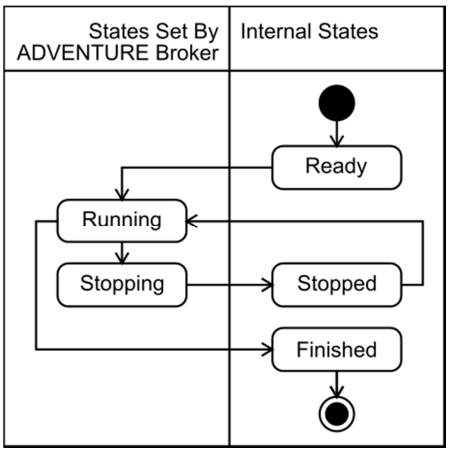

Figure 73 - Global Instance State

In the global instance state (Figure 73) there is a differentiation between two different kinds of states:

- Internal states that will be set during execution by the SPE.
- ADVENTURE broker settable states: Setting the state to “running” will be equivalent to start(), setting the state to “stopping” will be equivalent to stop().

The “stopped” state will be reached when the SPE is in a consistent state (e.g. no more running activities, open connections with gateways etc.), as the result of

- An ADVENTURE Broker setting the state to *stopping*.
- An error occurring in the control flow of the instance.

The “ready” state will only be available immediately after instantiating an instance before it is started the first time.

5.8.1.2 State Dependent Modifiability of Instances

Each instance will have all the attributes of the process (description, endpoints, data elements). For each instance these attributes will be modifiable only in *ready* and *stopped* states. In *finished* no changes to the instance will be allowed.

The changeability of the instance specific attributes will be dependent on several factors:

- *Position* will be
  - changed automatically during instance execution in states *running* and *stopping*, and
  - user changeable while in states *ready* and *stopped*.
- *State* will be changed automatically when in state *running* or *stopping*. For the internal states *ready* and *stopped*, the ADVENTURE is responsible for state changes
- *Status* will be changeable as part of the process logic in *running* state
- *Name* will be not modifiable at all

5.8.1.3 Simulation of Execution

As shown in Figure 72, starting a process instance will be either possible in simulation mode (simulation: 1) or standard mode (simulation: 0).

Simulation mode is defined as a special protocol for communicating with a set of complementary operations (i.e. provided through the gateway) that access Business Intelligence on top of a 3rd party system. These complementary operations are provided in the case that the 3rd party system is able to gather the appropriate data. For each standard operation it will be possible to defined means of simulating it: during Data Provisioning it will be possible to assign e.g. a "simulation operation" to a standard operation, which can be used to retrieve a set of meta data regarding the execution of this operation. Details about Simulation can be found in *Section Simulation and Forecasting* in this document. From the point of view of the process execution, besides implementing a different protocol, no additional changes will be necessary.

5.8.1.4 Process Structure

As shown in Figure 72 each process can consist of multiple elements. Two different kinds of elements are differentiated:

- Control Structures: Choose & Alternatives to realize In-/Exclusive[1] control statements, Parallel control statements (with Branches), Loops, and Critical Sections (Mutexes). These control structures are sufficient to capture the full expressiveness of e.g. BPEL as well as YAWL, as elaborated in [2]. As the process description created by the Process Designer will be translated into an intermediate executable language, this will allow for the continuous development of a specialized process description language in ADVENTURE.
- Nested Error Handling will be built into all constructs. E.g. for a loop it will be possible to specify a domain specific error handling strategy. The **group** element is a plain statement (without syntactic meaning) that will be usable as a container for error handling. Syntactic or unforeseeable errors will also be detectable by observing the events produced by the execution of an instance. Thus process independent error handling strategies might also be specified through the rule engine provided by the Monitoring component.
- Activities: The sole purpose of activities is to communicate with the gateways and the transformation services. In order to keep the process design and maintenance burden as low as possible, three different modes of communication will be realized that will be described in Section 5.8.1.6.

[1] Inclusive control statements allow for the execution of one or more conditional branches, Exclusive control statements allow for the execution of exactly on conditional branch. See e.g. BPMN notation.
[2] J. Mangler, G. Stuermer, and E. Schikuta, “Cloud Process Execution Engine-Evaluation of the Core Concepts,” Arxiv preprint arXiv:1003.3330, 2010.

5.8.1.5 Event Model & Event Driven Execution

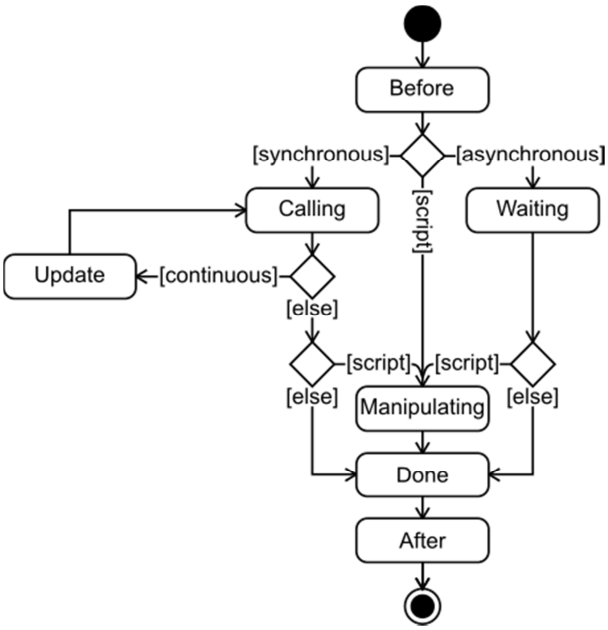

Figure 74 - Activity Level Events → Activity State

As shown in Figure 72 the SPE will rely on a comprehensive event mechanism to communicate its state to other components, e.g. Monitoring. The SPE will provide two different kinds of events:

- Instance Level Events which deal with changes that affect the whole instance. This includes position changes (activities that are currently in execution) as the process instance is executed, description / process model changes, state and status changes, as well as protocol changes (e.g. simulation vs. standard execution).
- Activity Level Events which occur while a single activity is executed. These events will realize a per-activity state model, as described in Figure 74.

Each activity can reach the following states:

- Before: a state that is reached by each activity, before any calls take place.
- Calling: for synchronous communication (when a gateway is called) the state of the activity switches to calling.
- Waiting: for asynchronous communication (process instance is waiting for a message from a specific gateway, with some specific message properties, the state of the activity switches to calling.
- Update: while in synchronous calling state, an activity will be able to continuously receive updates.
- Manipulating: After each communication with the gateway, the results possibly have to be put into or merged with existing data elements (process instance specific variables). It may be also necessary to modify *endpoints* or the *status*. While doing this the activity is in state *manipulating*. Alternatively if the task is a pure script task,

that modifies the data elements, endpoints or the status, it will also be in state *manipulating*. *Manipulating* will be an atomic operation.

- Done: After all communication and manipulating is performed, the activity will move to state *Done*.
- After: The artificial state *After* will be used as an intermediate state for signalling that an activity has successfully committed, but the start of the next activity has not yet been started.

Control structures as defined above will not yield any events, only activities.

5.8.1.6 Process: Micro vs. Macroflows

In order to facilitate the ADVENTURE Broker in supervising the virtual factory, two different types of processes will be used:

- Microflows are short-running process instances that typically do not require any human interaction, e.g. querying data from sensors or smart objects. Monitoring the status of such a process instance has little value, as they are executed and finished very quickly. Microflows will yield important information about the status of sensors and/or the status of a production and hold this information in their data elements. Whenever a microflow is in "finished" state, its "status" attribute is defined to point to the relevant information that it yielded:
    - Subsequent invocations for the same gateway operation and context (e.g. order status) may produce a table of historic data.
    - Calls to different contexts (e.g. querying status of different stock levels for different products in different partners) may be assembled into a stock level overview table, given the result of these calls is comparable.
- Macroflows are long-running process instances, which may include user interaction on the side of the partners (e.g. tool up machines, updating data in legacy systems). A typical case for such a process is an order. The duties of an ADVENTURE Broker include the monitoring of orders through the ADVENTURE platform. Thus the information about orders have to be presented in a up-to-date, coherent and lucid form. The SPE will support this monitoring by providing the real-time events described above to the Monitoring component which in turn will present them to the ADVENTURE Broker.

From the point of view of the SPE both the above flows with be handled the same. As stated above, the distinction between these two process types will affect how data from the virtual factory will be presented to ADVENTURE Brokers.

5.8.2 Use Cases

The SPE will be usable through the Dashboard in the following ways:

- To start / stop a *Main* process.
- To manually start / stop a *Standard process (e.g. due to a 'repair' situation, manual intervention).*
- To display a list of process instances and their *state* and *status*.

- Instantiating the execution of a PLUG process instance established in the Process Designer.

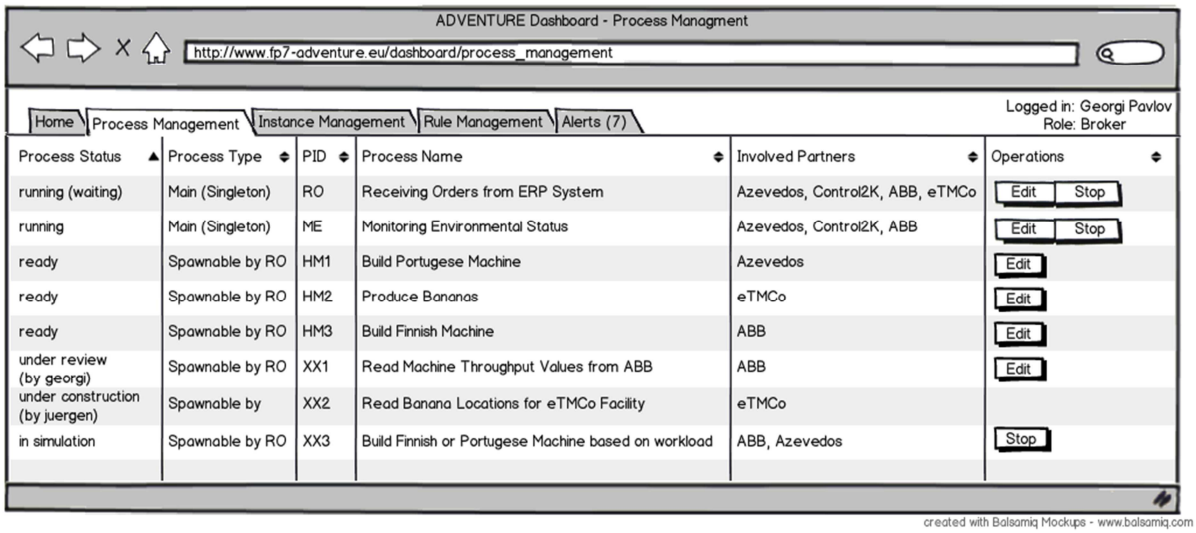

Figure 75 - Process Management Mockup

The mockup in

Figure 75 shows a list of processes. The Column “Process Status” is intended to show the status of a process. For Main processes (only instantiated once) it can show if it is running (and waiting for asynchronous events), while for standard processes it can show if they are ready for instantiation (either manually or by a Main process). This list shows also that some process models are currently worked on or in review. The Column “Process Type” will show if a process is of type “Main”, or if it is a “Standard” process that can be instantiated by a “Main” process.

Both columns contain values, explained in Section 5.8.

The rest of the columns are self-explanatory. The last column shows how the SPE and the Process Editor will work together:

- Processes (process models) can always be edited, unless they are currently simulated.
- Running “Main” processes and simulations can be stopped. Starting/Stopping simulations will be also exposed through the Process Designer.

The “Process Management” view will be realized as a collaboration between the Process Design Component and the SPE, as the information in the table mostly stems from the process model, whereas possible operations are determined by either the SPE or the Process Designer. For example stopping is only possible for running Main processes.

Figure 76 shows the management of running process instances. As shown in column “Monitoring” a drill-down into the details of particular instances (through the Monitoring Component) will be possible.

This view will be realized through the SPE component and will provide:

- A summarized view of all current running instances
- A means to search and filter running instances
- A means to stop running instances, as well start instances that are ready or stopped
- A means to invoke the Process Editor for changing instances that are currently stopped (in case they need structural repair). Please note that instances that are currently in state "*stopping*" are not editable yet.

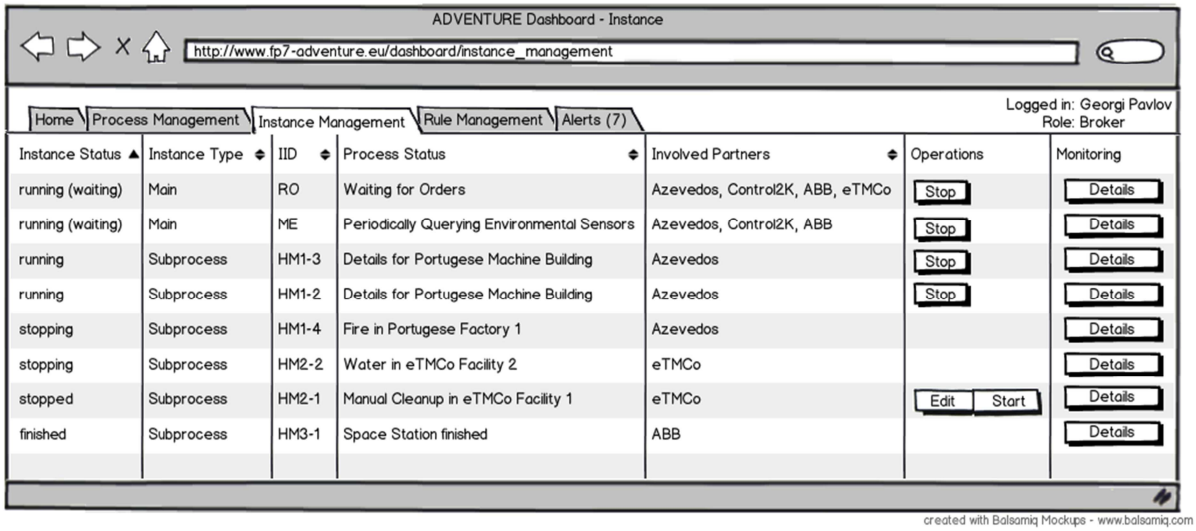

Figure 76 - Instance Management Mockup

Figure 77 gives an overview of the use cases involved in realizing these mockups. This involves the interaction of the user with the system, as well as the interaction between components of the system.

It has to be noted that finished process instances may be shown in this table as well, as directly exploring e.g. old orders may be beneficial to the ADVENTURE Broker. As they may clutter the list, a means for hiding them will be provided.

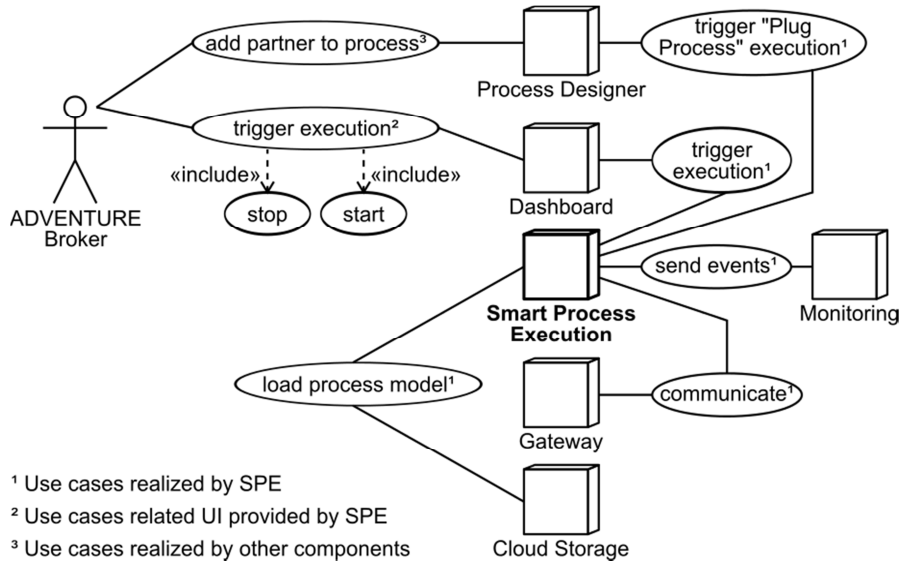

Figure 77 - SPE Use Cases

Use Case: SPE-U01a	**Name: trigger execution (start)**
Description	The SPE will provide a simple REST (update) operation to instantiate a process from an existing process model. For already instantiated but stopped instances, execution will resume.
Actors	ADVENTURE Broker
Pre-conditions	The process is syntactically correct and all activities are connected to existing gateways and/or transformation services.
Inputs	Process ID or Instance ID

Events sequence	• For Process ID:
	  • ADVENTURE Broker selects process and presses “START” button
	  • Dashboard sends Process ID to SPE
	  • Process model is loaded from Cloud Storage.
	  • SPE creates an Instance from the model
	  • The instance is executed
	• For Instance ID:
	  • ADVENTURE Broker selects instance and presses “START” button
	  • Dashboard sends Instance ID to SPE
	  • SPE selects instance and executes it.

Extension points	N/A
Outputs	Instance ID
Post-conditions	Instance is in state “running”
Requirements map	N/A
Related Mock up	Figure 75 - Process Management Mockup Figure 76 - Instance Management Mockup

Use Case: SPE-U01b	Name: trigger execution (start - Main)
Description	Similar to SPE-U01a, but for special process type “Main” that can be only started once. E.g. a process that waits for messages from an ERP system.
Actors	ADVENTURE Broker
Pre-conditions	The process is syntactically correct and all activities are connected to existing gateways and/or transformation services. A process of type “Main”. The process has no running instance yet.
Inputs	Process ID
Events sequence	• ADVENTURE Broker selects process and presses “START” button • Dashboard sends Process ID to SPE • Process model is loaded from Cloud Storage. • SPE creates an Instance from the model • The instance is executed
Extension Points	N/A
Outputs	Instance ID
Post-conditions	Instance is in state “running”
Requirements map	N/A
Related Mock up	Figure 75 - Process Management Mockup Figure 76 - Instance Management Mockup

Use Case: SPE-U02	Name: trigger execution (stop)
Description	Stop a currently running process instance.
Actors	ADVENTURE Broker
Pre-conditions	A running process instance.
Inputs	Process Instance ID.
Events sequence	• User selects a running process instance and presses “STOP” button • Dashboard sends Instance ID to SPE • SPE selects the Instance and puts it into “stopping” state. • When a consistent state is reached the SPE is going to “stopped” state.
Extension Points	N/A
Outputs	N/A
Post-conditions	Process instance is in state “stopping”.
Requirements map	N/A
Related Mock up	Figure 75 - Process Management Mockup Figure 76 - Instance Management Mockup

Use Case: SPE-U03	Name: load process model
Description	Load a process model or a process instance model.
Actors	SPE
Pre-conditions	N/A
Inputs	Process ID or Process Instance ID.

Events sequence	• The SPE accesses the Cloud Storage to ...
	  • Load a Process Model in case a new instantiation.
	  • Load a Process Instance Model in case a modified process instance is to be resumed.

Extension Points	N/A
Outputs	Process Model.
Post-conditions	N/A
Requirements map	N/A
Related Mock up	N/A

Use Case: SPE-U04	Name: trigger “Plug Process” execution
Description	A Plug Process is executed whenever a partner is added to a particular process through the Process Designer. Each partner has its own Plug Process that is specified during the Data Provisioning phase. The purpose of the Plug Process is to facilitate data exchange between Virtual Factory partners to help all parties decide if adding a partner is feasible.
Actors	Process Designer Component, ADVENTURE Broker
Pre-conditions	A process model currently edited.
Inputs	Process Instance ID.

Events sequence	• User selects a potential partner and presses the “Plug” button.
	• The SPE is triggered, loads the Plug Process for that partner, instantiates and executes it.
	• The ADVENTURE Broker has to provide certain information requested by the partner (i.e. quantitative and qualitative goals).
	• The SPE returns the results of the Plug Process instance to the ADVENTURE Broker. The results may include:
	  • FALSE: The partner refuses to be added to this process du to internal reasons (i.e. no capacity left).
	  • TRUE**:** The partner is ready to be added and provides additional context specific dynamic information. Based on this information the ADVENTURE Broker may decide if this partner will be added.

Extension Points	N/A
Outputs	Status of the Plug Process.

Post-conditions	Plug Process instance is in state "*finished*".
Requirements map	F2
Related Mock up	N/A

Use Case: SPE-U05	**Name: send events**
Description	Send Process Instance execution and maintenance events to the Monitoring component as described in Section 5.8.1.
Actors	SPE Component, Monitoring Component
Pre-conditions	Active Monitoring Component.
Inputs	N/A.
Events sequence	SPE fires off an event whenever something happens to an instance (see above).
Extension points	N/A
Outputs	Event.
Post-conditions	N/A
Requirements map	N/A
Related Mock up	N/A

Use Case: SPE-U06	**Name: communicate (with Gateway component)**
Description	While activities are executed as explained in Section 5.8.1., the operations on the gateway are invoked as described in Section 5.7.1.
Actors	SPE Component, Gateway Component
Pre-conditions	Active Gateway.
Inputs	Parameters provided through the control flow.
Events sequence	Calling, Waiting and/or receiving status updates from the gateway.
Extension points	N/A
Outputs	Data returned by the gateway.
Post-conditions	N/A
Requirements map	N/A
Related Mock up	N/A

5.8.3 Components Interaction

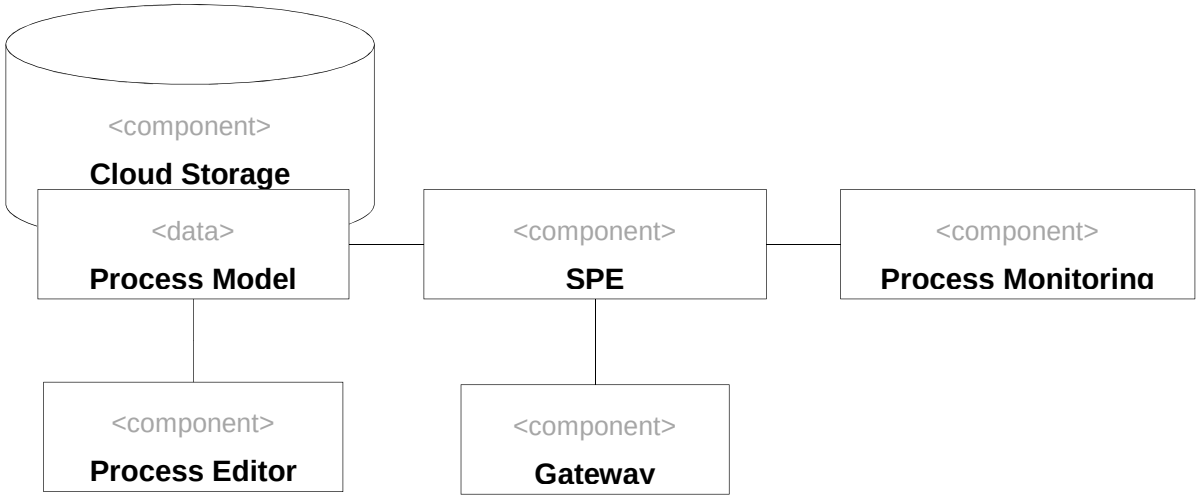

Figure 78 - SPE Component Interaction

As already elaborated in Section 5.8.2, the SPE will utilize Process Models produced by the Process Designer to coordinate / route all interactions between ADVENTURE partners through the gateways. While doing so, the SPE will produce a series of events as described above. This section will give details about:

- Different modes of interaction with the gateway
- Means of shaping the process execution through the rule engine provided by Process Monitoring

5.8.3.1 *Interaction through A Gateway, Correlation*

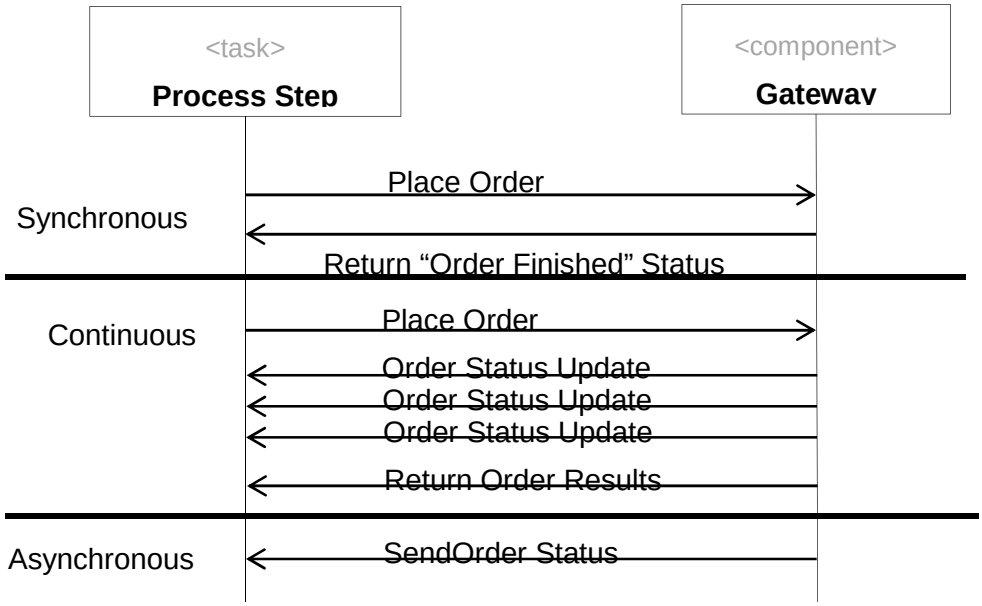

Figure 79 - Message Interaction Patterns

As already depicted in Figure 79 there will be several means of interaction through a gateway:

- Synchronous: a request is sent to the gateway, the gateway sends a response. For this case the structure of the message has not to be introspected. The gateway can respond, using a unique connection ID, provided with each request to a gateway.
- Continuous: A sub-case of *Synchronous* communication. Between the request and the response, several status updates can be sent, which will be made available to monitoring as described above. For the purpose of sending the status updates, in addition to the unique connection ID, a status update ID will be provided, which will be renewed after each status update.
- Asynchronous: This case will be realized through a technique called correlation[3]. Messages have to be analysed based on certain criteria (e.g. order-id) to determine which message belongs to which activity / instance. This case occurs, whenever the gateway is not actively fetching values, but receives push messages from legacy systems. Thus this case may also apply to the Continuous case, when waiting for updates.

In order to support the Asynchronous case, the Process Designer has to support the specification of correlation. E.g. for a given activity that is waiting for a message, the

[3] A. Barros, G. Decker, M. Dumas, and F. Weber, "Correlation Patterns in Service-Oriented Architectures," in Fundamental Approaches to Software Engineering, vol. 4422, M. Dwyer and A. Lopes, Eds. Springer Berlin / Heidelberg, 2007, pp. 245–259.

ADVENTURE Broker has to specify that the message has to contain e.g. the specific Order ID for the Order that is dealt with in this particular Process Instance. In order to do so, the Order ID has to be present in a data element inside the instance, and the waiting activity has to be restricted to wait for messages (of a certain structure) that contain the same Order ID as saved in the data element.

5.8.3.2 Process Execution Shaping

Whenever a process is executed, a set of events will be sent to Monitoring components present in the system. Instead of always sending all events, a Monitoring component will be able to subscribe to a sub-set of the events. Additionally it will be possible for a Monitoring component to reply to events and thus shape the execution of a process. This will be particularly useful for the Rule Engine which is part of the ADVENTURE Monitoring component.

The events “before” and “after” as described above will wait for a response. The response will be either:

- “continue with the execution”
- “please wait for later decision”

Thus the monitoring, when detecting a certain condition through the rule engine, can suspend execution until a decision is reached. Possible decisions might include:

- To continue with the execution
- To delay the execution while other processes are instantiated and executed
- To stop the execution and notify an ADVENTURE Broker

5.8.4 Conceptual Data Model

The key domain model elements defining the SPE are Processes and Instances. All other concepts are related to these elements and have been described in detail in section 5.8.1.

5.8.5 Parameters To Take Into Account For Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+++
Regularly Updated	+
Technical Up-to-Dateness / Appeal	++
Open Source	+
Non-Infecting	++
Code-Quality	+
Extensibility	+++
Community	+
Performance / Scaleability	++
Reuse of existing developments	++
EU project origin	--
Platform (Portability)	+++
Open Standards Compliance	+
Interoperability (easy integration for all platforms)	+++
Specific Parameters	
Reliable Messaging / Receipt Acknowledgement	+++
Provide binary message exchange	++
Secure communication protocol	+++
Signed messages	+/-
Provide message buffering	+++
Point-to-Point Messaging for Component Instances	+++
Ability to treat users as message partners	++
Multi-recipient-messaging	++
Multi-instance-Server for cloud hosting	++
Provide configuration UI for routing	+
Provide user-message UI	-
Message Partner Presence Awareness	+
Uses UFT-8	+++
Lightweight Infrastructure for Remote SME Endpoints	++
Communication Partner Registry included	+

5.9 Process Adaption

The Adaptation Component (AC) will be an independent service with the purpose to handle all runtime aspects regarding the adaptation of process models.

5.9.1 Overall Functional Characterization

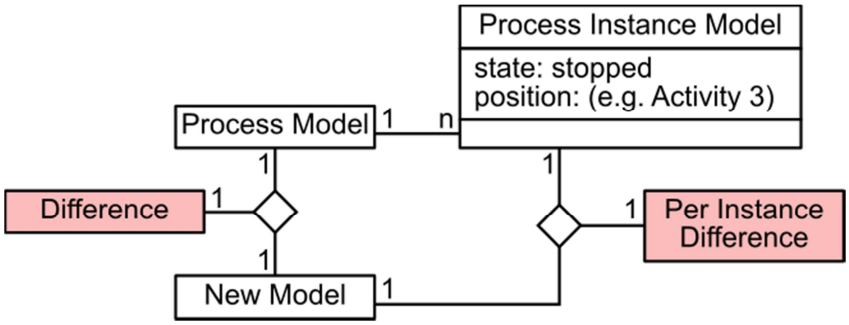

Figure 80 - Adaptation Component (AC) Concepts and their Dependencies

The main purpose of the Adaptation Component will be to deal with the application of optimization or in the most general sense – changes - to process models and process instance models. Changes to processes may occur because of the following reasons:

- Through a simulation and subsequent optimization, an ADVENTURE Broker has created an improved version of a process
- Because some partners operations change e.g. become too expensive, or can no longer deliver, a process has to operate with a new set of partners
- Because of an error in a process instance, the instance has to be manually repaired (e.g. insert some additional activities)

So in general there are two sources for New (Process) Models (see Figure 80 above):

- When a user manually creates an evolution of a model
- When a user runs an optimization and gets a new (better model)

For any given process type, there are typically many instances in various states, meaning some are close to finishing; some are just at the beginning. As these processes may take days, assigning a change just to the process model that spawns new instances may lead to a situation where running instances exist that are representing several different evolutions. Additionally single instances may further deviate due to manual repair actions.

It is in the best interest of the an ADVENTURE Broker that as many running instances as possible are also migrated to the new model:

- For some instances this may be possible as the changes are behind the current position of execution
- For some instances this may be partially possible
- For some instances this may be not possible

ADVENTURE will pursue the following approach - whenever an ADVENTURE Broker wants to apply a new version of a process model:

- To a process instance: the difference is calculated and applied if possible
- To a process: the difference for the process model and each process instance model is calculated and not only is the process model migrated, but as many process instances as possible

The worst case in the second scenario would be the same as having no adaptation at all. In the average case it will lead to improved manageability of Virtual Factories, as less different process versions exist.

5.9.2 Use Cases

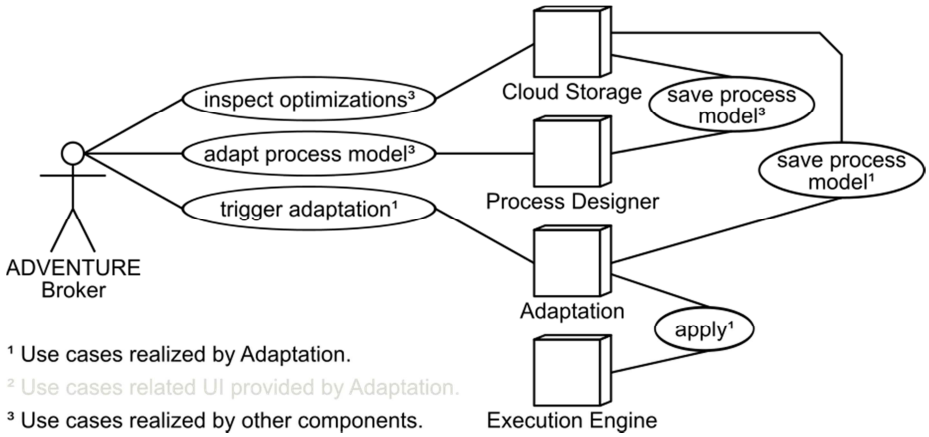

Figure 81 - Adaptation Use Cases

Use Case: AC-U01	**Name: trigger adaptation**
Description	Adaptation can be triggered by an ADVENTURE Broker after he decided for a particular Process Model Evolution (stemming from Optimization, manual Adaption). Adaptation can be triggered for single stopped instances or for whole processes (process models and process instance models).
Actors	ADVENTURE Broker
Pre-conditions	There is a computable difference between process model and process models that should be changed.
Inputs	Process ID or Instance ID; ID of NEW Process Model

Events sequence	• For Process ID:
	  • Apply change to Process Model, and save to cloud storage.
	  • Calculate difference to all instances.
	  • If change is applyable, stop instance, apply and save new models to cloud storage.
	  • Continue modified instances.

	• For Instance ID:
	  • Check if instance is stopped.
	  • Calculate difference for instance.
	  • If change is applyable, apply and save new model to cloud storage.

Extension points	N/A
Outputs	A list of successful / failed Instances and Processes.
Post-conditions	Modified process instances
Requirements map	F31
Related Mock up	N/A

Use Case: AC-U02	**Name: save process model**
Description	Whenever a change applicable, save the result to cloud storage. Occurs as part of AC-U01.
Actors	Adaptation Component.
Pre-conditions	A change is applicable to a particular Process / Process Instance model.
Inputs	Process ID or Instance ID
Events sequence	Save model.
Extension points	N/A
Outputs	Success / Failure
Post-conditions	Next version of Process / Process Instance Model is available through cloud storage. Alternative: Process Instance in SPE is updated.
Requirements map	N/A
Related Mock up	N/A

Use Case: AC-U03	**Name: Apply**
Description	Whenever a change is applicable for a particular instance, change the representation in the SPE. Occurs as part of AC-U01.
Actors	Adaptation Component.
Pre-conditions	A change is applicable to a particular Process Instance Model.
Inputs	Instance ID
Events sequence	• [Optional] Stop instance. • Apply changes to instance. • [Optional] Continue instance.
Extension points	N/A
Outputs	N/A
Post-conditions	Changed Process Instance.
Requirements map	N/A
Related Mock up	N/A

5.9.3 Modules Interaction

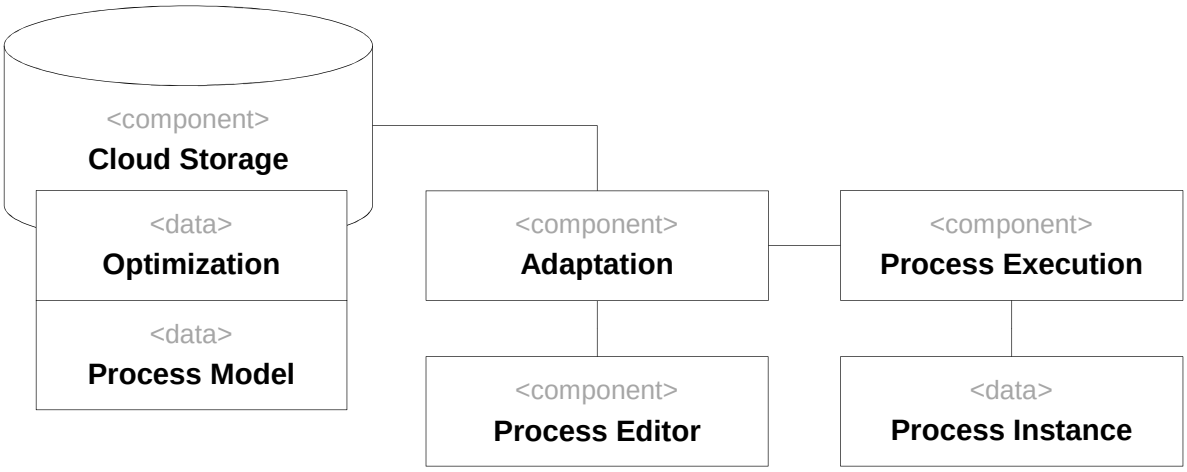

Figure 82 - Component Interaction

The Adaptation component reads either optimal solutions (provided through Optimization) or manually changed Process Models from the Cloud Storage. It usually does not interact with other components, unless it is triggered by them or interacts with the Process Execution. AC may be triggered through the Process Designer.

Figure 82 shows interactions between the AC and other components.

5.9.4 Conceptual Data Model

The key domain model elements defining the AC are calculated Differences to either Process Models or Process Instance Models. All other concepts are related to these elements and have been described in detail in Section 5.9.1.

5.9.5 Parameters To Take Into Account For Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+/-
Regularly Updated	+/-
Technical Up-to-Dateness / Appeal	+++
Open Source	+/-
Non-Infecting	+/-
Code-Quality	+/-
Extensibility	---
Community	---
Performance	--
Reuse of existing developments	+/-
EU project origin	+/-
Platform (Portability)	---
Open Standards Compliance	+/-
Interoperability	+++
Specific Parameters	
Differences between process / process instance models	+++
Application of changes to partially executed process instances	+++

5.10 Process Forecasting and Simulation

The Forecasting & Simulation (FS) component will be an independent service that relies on the Process Execution component and uses gateways to get temporal and execution information from partners without e.g. ordering something or invoking partner processes. Simulation data is always connected to particular versions of a process model and:

- Is intended to help ADVENTURE Brokers to decide for / against particular partners during design processes.
- Will serve as input for process optimization.

5.10.1 Overall Functional Characterization

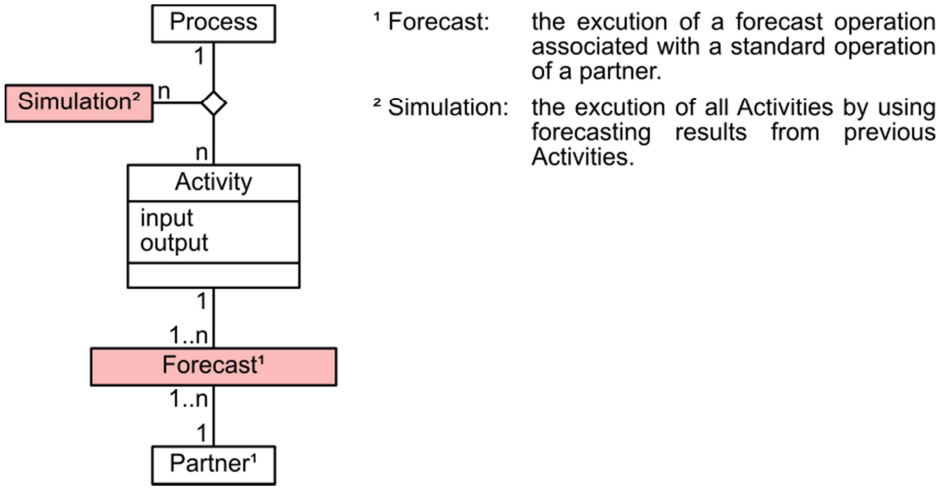

Figure 83 - FS Concepts and their Dependencies

As shown in Figure 83, the basis for the FS component will be ADVENTURE processes (explained in detail in the Process Execution Component Section). These processes consist of activities, which are associated with one or more operations in 3rd party systems (multiple interchangeable partners might provide a certain operation), which are exposed through an ADVENTURE gateway.

The idea of the FS component is that while an ADVENTURE Broker is using the Process Designer, they needs detailed information about how a certain partner / combination of partners may behave. As a newly designed process is typically unique and strongly depends on the semantic context (i.e. the products it produces) of the factory, this information can only be delivered by the partner.

Simulation in ADVENTURE is defined as the instantiation and execution of a particular process, not with the goal of e.g. ordering a product, but with the goal of collecting information:

- How long may particular activities in this process take?
- What range of possible results may they return?

In contrast to e.g. a normal order, the results are intended to be returned instantaneously, by invoking not a normal order operation, but a different operation associated with the order operation that is connected to some kind of Business Intelligence logic.

In the context of ADVENTURE, Forecast is defined as being the invocation of this Business Intelligence (BI) logic for a single partner.

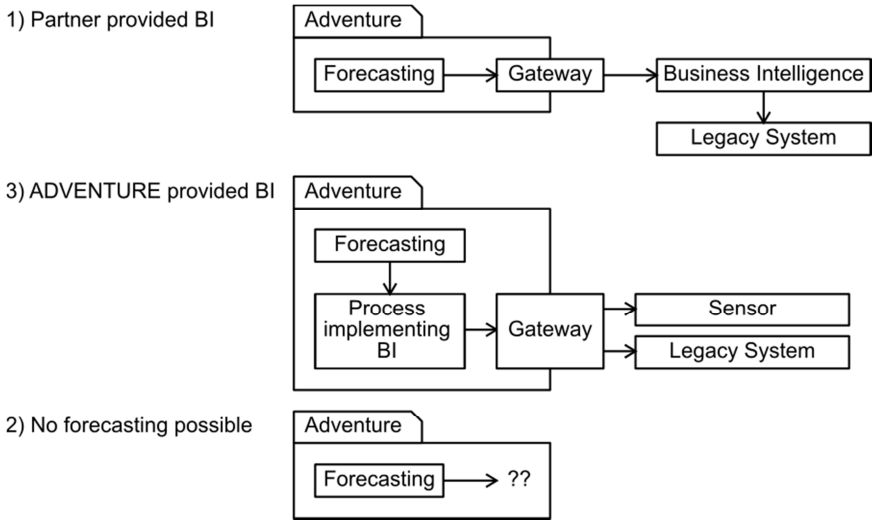

Figure 84 - Three Possible Ways of Forecasting

As depicted in Figure 84, ADVENTURE will support 3 ways of Forecasting:

- Forecasting on the basis of invoking BI logic provided by a partner. This BI logic will most probably be integrated with a particular partner's legacy systems and could provide very accurate forecasts.
- Forecasting on the basis of instantiating an ADVENTURE process that implements BI logic, by e.g. querying sensor data or stock levels from a particular partner. This BI process may be either designed by the ADVENTURE Broker of a particular Virtual Factory, or provided by the partner (ADVENTURE member) during Data Provisioning.
- Forecasting is not possible, as none of the above alternatives exist. In this case simulations may be only partially possible.

5.10.1.1 Forecasting

ADVENTURE defines forecasting as the invocation of a special Business Intelligence operation, which may be provided in conjunction with a normal operation, in order to collect data about the expected behaviour of the normal operation. ADVENTURE defines forecasting information to consist of data delivered by the partner, in contrast to historic data collected by the Monitoring Component inside an ADVENTURE virtual factory. Forecasting information may be provided by the partner by either:

- considering factors like workload or stock levels, or
- by return static experience-based values.

As stated above, forecasting operations may either be directly available by partners, or point to an ADVENTURE process to be executed in order to get results.

All ADVENTURE forecasting operations are intended to have a standard interface consisting of two parts, containing the following information:

- Part 1 – Quantitative information: The predicted optimistic (0), normal (M) and pessimistic (P) duration, given a particular set of input parameters as a side condition.
- Part 2 – Qualitative information: A set (array) of predicted possible outputs from a task, weighted by the possibilities under which they might occur. Activities in a process occur in sequences, thus they rely on particular input possibly influenced by previous invocations of activities. As activities all provide a different output, the elements contained in the result array will all have the structure defined for a particular activity. Example: an activity is the representation of "ordering a chair", the result of this task may consist of:
    - Attribute "Status": OK
    - Attribute "Quality": A+
    - Attribute "Material Testing": ISO 17025 certified

For this case a forecast will include an array of cases that might occur, including the frequency predicted by the partner (note that "Material Testing" will probably remain constant across all possibilities).

While quantitative information can and will be made directly available to optimisation to e.g. calculate float, slack, the critical path, and highlight critical activities[4], qualitative information is more difficult to acquire: as multiple value sets may be returned by each activity and these value sets may be defined to serve as input for other activities in the process, a large number of possible combinations will arise for subsequent activities. This brings us to simulation.

5.10.1.2 Simulation

The Simulation Component will handle two main tasks:

- The efficient coordination of calling each activity in a certain process with all possible combinations of input parameters that will arise.
- The preparation of a set of annotated process models, each covering a particular combination of input/output parameters that may occur during an execution, including probabilities and quantitative values.

Thus the Simulation Component will produce a full enumeration of scenarios that might occur, given a certain set of partners. This set of scenarios (annotated process models utilizing certain partner combinations) will serve as input for the Optimization Component. The purpose of Optimization is to select a particular "optimal" scenario, based on goal functions defined by the ADVENTURE Broker.

A simulation may be triggered either by an ADVENTURE Broker (while editing a process) or through the Optimization Component to acquire data.

[4] As defined through the PERT (Program Evaluation and Review Technique).

In order to further exemplify simulation, Figure 85 shows a basic overview of the simulation algorithm when it is triggered. After reading and instantiating a process (model) it will start the execution of a process by using a special protocol, which is able to:

- Identify and call the Business Intelligence (forecasting operations) instead of normal operations
- Handle the results that differ from normal invocation

After acquiring a forecast, simulation will then analyse the results, create new instances of the same process, forward to the previously executed activity, making it believe that one particular element of the result array has been returned, and continue execution from this point. This way a large set of instances representing possible process execution traces will be generated, which in the end have to be analysed to create the set of possible scenarios.

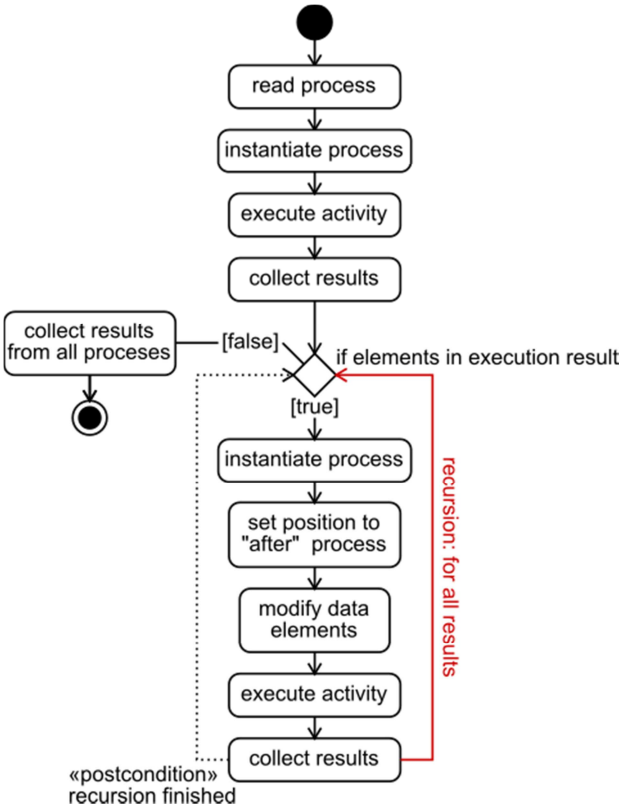

Figure 85 - Basic Algorithm

While manually triggering a simulation, an ADVENTURE Broker will be able to observe the progress of the simulation, as well as receive a detailed report covering all scenarios.

Note: The Monitoring Component will provide a similar view, but instead use live and historic data.

Providing continuous simulation for all processes, even while designing (without manual activation) while be a low priority. While it would possibly increase usability a lot, it can be predicted that it will be only feasible after a thorough performance optimization of the basic algorithm shown in Figure 85.

5.10.2 Uses Cases

Most use cases described in these sections appear integrated into other components, thus separate UI's are only provided to show progress or reports. For the Process Editor a simple "start simulation" button is sufficient.

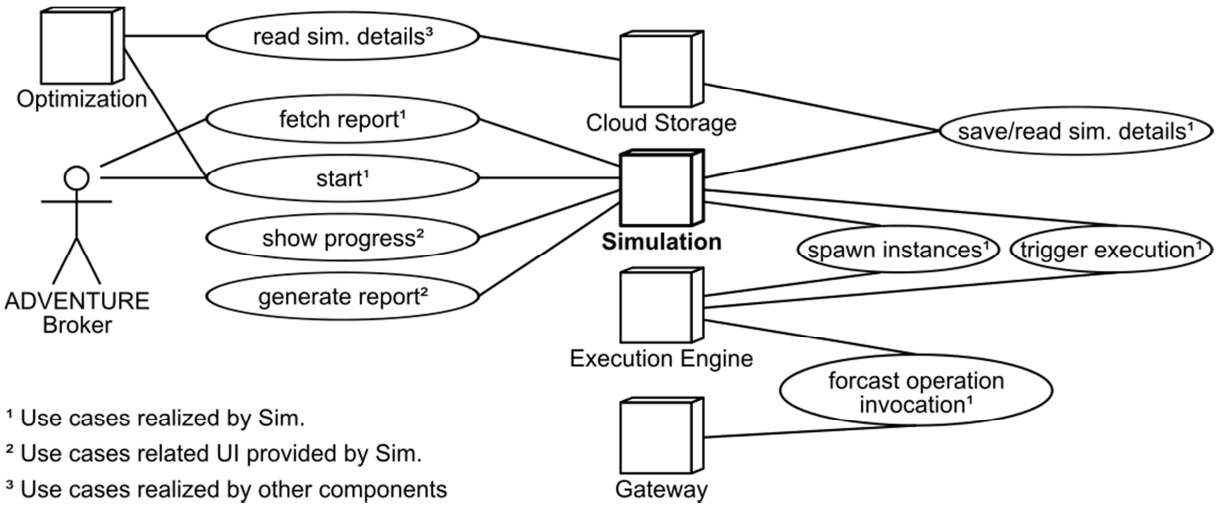

Figure 86 - Forecasting & Simulation Use Cases

Use Case: SF-U01	**Name: start**
Description	Starting a Simulation is triggered for a particular Process or Process Instance (the process model of a process instance can deviate from the process model it has been instantiated from, do to ad-hoc repair).
Actors	ADVENTURE Broker
Pre-conditions	• Process or Process Instance Model is saved and in a consistent state in the Cloud Storage. • For each activity of the model, one or more potential partners (respectively operations in the gateway of the partner) have been assigned. • No other simulation process is running
Inputs	Process ID or Instance ID
Events sequence	• Trigger Start of Simulation. • For details how the simulation will work, see Figure 85 above.
Extension Points	N/A
Outputs	N/A
Post-conditions	A set of running Simulation Process Instances in the SPE.
Requirements map	F39, F46, F81+ F82 (depends on the process and activities for which simulation is run).
Related Mockup	N/A

Use Case: SF-U02	Name: Fetch report
Description	After a simulation is finished, an ADVENTURE Broker will be able to fetch a report for this particular analysis. As the report will contain all possible combinations of interactions with partners, what-if analysis will be possible. Fetching reports for historic simulations will not be possible.
Actors	ADVENTURE Broker
Pre-conditions	All activities for a particular process or process instance model have been queried in forecasting mode.
Inputs	Process ID or Instance ID
Events sequence	• Send request to fetch report. • Read simulation details from Cloud Storage. • Generate report. • Return report.
Extension Points	N/A
Outputs	Report (HTML Document)
Post-conditions	N/A
Requirements map	F25, F26, F28, F40, F44, F45
Related Mockup	N/A

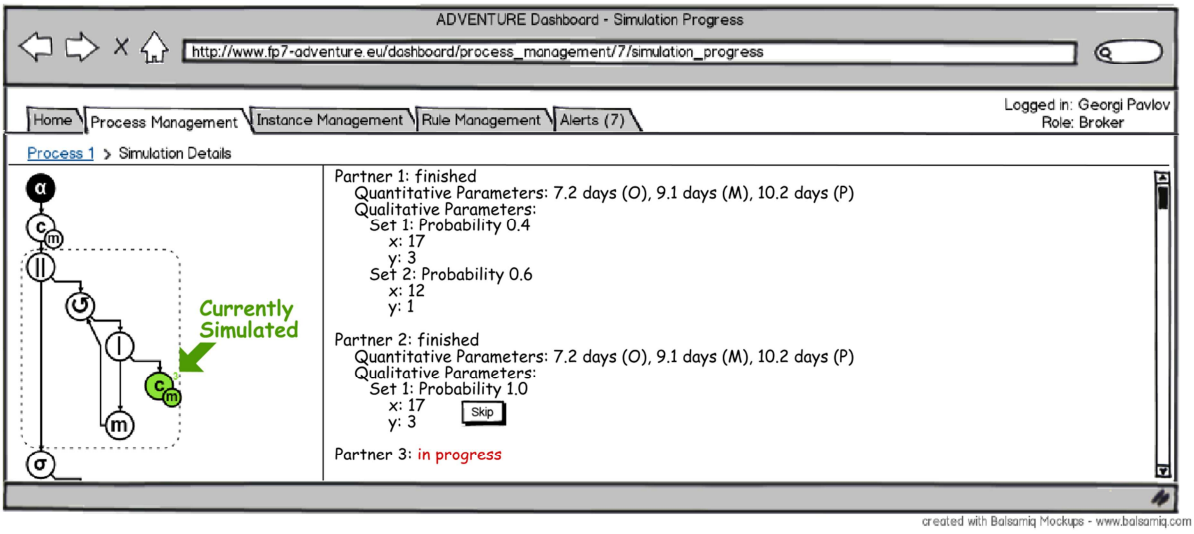

Figure 87 - Simulation Progress / Details Mockup

Use Case: SF-U03	Name: Show progress
Description	Show the progress of the simulation, with the purpose of giving the user to ability to skip certain partners for certain activities to avoid situations where simulations take forever, because certain partners handle forecast manually.
Actors	ADVENTURE Broker
Pre-conditions	Running Simulation for a particular process model or process instance model.
Inputs	Process ID or Instance ID

Events sequence	• View live progress in graphical representation of Simulation. • Skip partners for “blocked” activities.
Extension Points	N/A
Outputs	Monitoring Interface
Post-conditions	N/A
Requirements map	N/A
Related Mockup	Figure 87 - Simulation Progress / Details Mockup

Use Case: SF-U04	**Name: generate report**
Description	Generate the current report, when requested by an ADVENTURE Broker.
Actors	Simulation Component
Pre-conditions	Request by an ADVENTURE Broker
Inputs	Process ID or Instance ID
Events sequence	• Read simulation details from Cloud Storage. • Generate Graphical representation of the associated Process (Instance) Model. • Annotate graphical representation for each result scenario (a particular combination of partners, see above).
Extension Points	N/A
Outputs	HTML Document.
Post-conditions	N/A
Requirements map	F44
Related Mockup	N/A

Use Case: SF-U05	**Name: save/read simulation details**
Description	After a simulation has been finished it has to be written to the cloud storage.
Actors	Simulation Component
Pre-conditions	N/A
Inputs	Process ID or Instance ID
Events sequence	For a particular simulation (process model or process instance model), the simulation results are saved to cloud storage
Extension Points	N/A
Outputs	Link to Simulation Details
Post-conditions	A set of running Simulation Instances
Requirements map	N/A
Related Mockup	N/A

Use Case: SF-U06	Name: spawn instances
Description	As described in Figure 85, the Simulation Component will spawn multiple instances during a simulation
Actors	Simulation Component.
Pre-conditions	A simulation is running.
Inputs	Process ID or Instance ID
Events sequence	• Simulation component takes the process model associated with an process or process instance, and uses it to instantiate a new instance. It does so for every results of every forecast.
Extension Points	N/A
Outputs	New Instance
Post-conditions	A new process instance in state stopped.
Requirements map	N/A
Related Mockup	N/A

Use Case: SF-U07	Name: trigger execution
Description	Simulation uses a previously spawned process instance (SF-U06), modifies it and starts it. As exemplified in in Figure 85, the purpose is to create one instance for every combination of forecasting results, as described in above.
Actors	Simulation Component
Pre-conditions	Stopped Instance
Inputs	Instance ID
Events sequence	• Set position to previously executed instance. • Trick the instance into thinking, one particular result element (of the array of qualitative results) has been return. • Start execution after the particular activity.
Extension Points	N/A
Outputs	A stream of events to be collected by the Simulation Component.
Post-conditions	A running process instance.
Requirements map	N/A
Related Mockup	N/A

Use Case: SF-U08	Name: Forecast operation invocation
Description	The SPE will contain a special Simulation Protocol Extension (Service), which instead of invoking a particular operation will invoke some Business Intelligence logic as described in Section 5.10.1
Actors	Simulation Component
Pre-conditions	Forecasting operation (Business Intelligence Logic) assigned to normal operation.
Inputs	As defined for a particular Activity.
Events sequence	• Set position to previously executed instance.

	• Trick the instance into thinking, one particular result element (of the array of qualitative results) has been return. • Start execution after the particular activity.
Extension Points	N/A
Outputs	As defined in Section 5.10.1.1.
Post-conditions	N/A
Requirements map	N/A
Related Mockup	N/A

5.10.3 Component Interaction

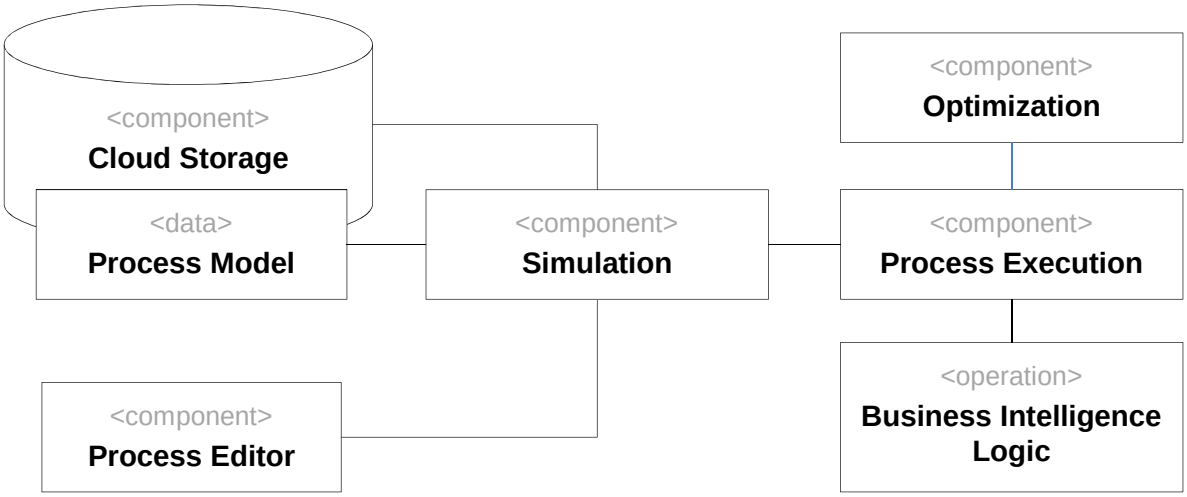

Figure 88 - Forecasting & Simulation Components

As depicted in Figure 88 and Figure 86 the following interaction with the simulation component is possible:

- An ADVENTURE Broker designs a process.
- An ADVENTURE Broker triggers a simulation (through the Process Editor) to get information about a certain set of partners.
- The Simulation Component uses the SPE to collect forecasting information.
- The Simulation Component saves all collected data to the cloud storage.
- The Simulation Component signals that a report is ready, and makes it available (a link anywhere in the dashboard / process editor can be shown).
- Optimization may use the collected data to come up with optimal solutions regarding certain goal functions.

A Simulation Progress UI will be provided so that it can be embedded e.g. in the process editor. The simulation results will be available in the cloud storage in a format that will allow reading forecasting details for single activities.

5.10.4 Conceptual Data Model

The key domain model elements defining the FC are processes and activities are:

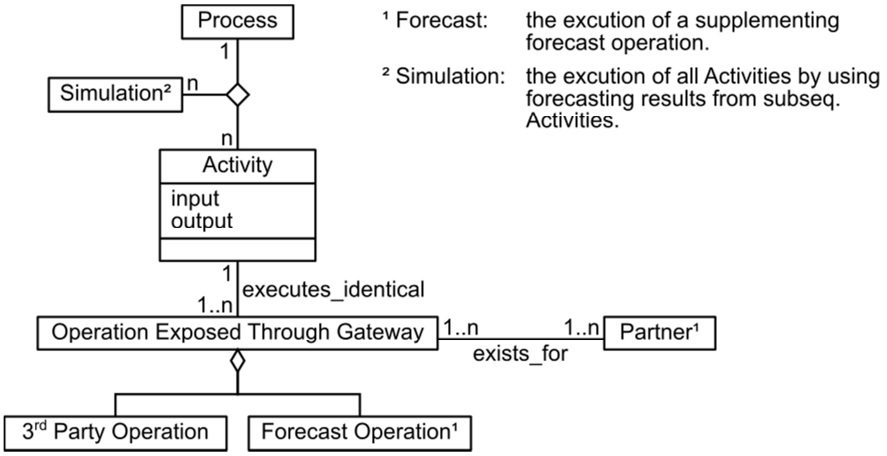

Figure 89 - Conceptual Data Model

As shown in Figure 89, the following concepts are involved:

- A simulation happens for a particular process, with a process being either:
  - A process model that is used to instantiate new processes
  - A process instance model that may potentially deviate from the process model it has been instantiated from due to structural repair actions
- Each process contains multiple activities, which in ADVENTURE represent interactions with a particular 3rd party operation exposed through a Gateway, with a given set of input and output parameters
- Each activity will be potentially connected to one or more identical[5] operations from different Partners
- To each 3rd party operation, a Forecast Operation will be assigned, which will trigger Business Intelligence logic

[5] How to ensure that different partners expose operations that are interchangeable, either through applying semantic reasoning or static transformations defined by the ADVENTURE Broker is to be resolved as part of the ADVENTURE research effort.

5.10.5 Parameters To Take Into Account For Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+/-
Regularly Updated	+/-
Technical Up-to-Dateness / Appeal	+++
Open Source	+/-
Non-Infecting	+/-
Code-Quality	+/-
Extensibility	---
Community	---
Performance	+++
Reuse of existing developments	+/-
EU project origin	+/-
Platform (Portability)	---
Open Standards Compliance	+/-
Interoperability	+++
Specific Parameters	
3rd Business Intelligence	++
Business Intelligence Process	+++
Forecast Data Format	+++
Simulation Algorithm	+++

5.11 Process Monitoring

The monitoring component in the ADVENTURE platform is the component that provides the real time monitoring of ongoing process, historical data relating to finished processes and instances and business analytics relating to process and activities types.

5.11.1 Overall Functional Characterization

The Process Monitoring component provides real time, log and performance data relating the virtual factory processes. As represented in Figure 90, it will contain 5 main sub-components:

- Monitoring Engine
- Real-time Monitoring
- Process Log
- Process Analytics
- Monitoring Rules Engine

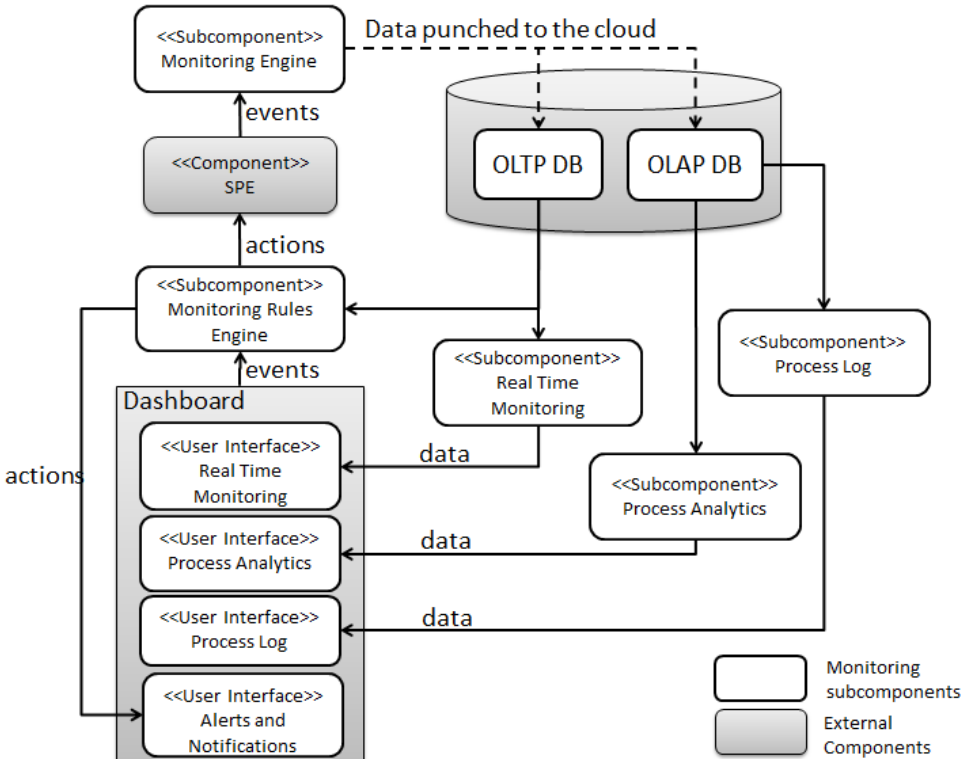

Figure 90 - Process Monitoring Subcomponents

The Monitoring Engine captures the events produced by the Smart Process Engine and stores the relevant event data in the cloud.

The Real-time Monitoring component provides a live view of the ongoing processes using the process editor interface, so that virtual factories brokers may decide to undertake flow

adjustments and efficient decisions in order to improve the performance of the manufacturing processes. The Process log component allows users to search for finished process instances and visualize its data in a graphical interface. The Process Analytics component provides the key performance indicators relating to the manufacturing processes and ADVENTURE partners. Finally, the alerts and notifications component allows the definition of rules based on process execution delays (e.g. if a process has 5 days to execute and it is now day 6) that are evaluated by the Rules Engine. This component throws alerts to the Dashboard, as well as performs action upon the Smart Process Engine.

5.11.2 Use Cases

The monitoring component has 4 use case packages. For each package, a use case diagram and the corresponding use case descriptions and interface mockups are presented in the subsections.

5.11.2.1 Real Time Monitoring Use Cases

ADVENTURE will provide real time monitoring of the on-going manufacturing processes. Three different data sources will be considered in order to show updated information to the user: process execution data (from smart process engine), smart objects data and legacy systems data. All this information will be available in a graphical user interface (global view). By clicking on specific tasks, or by access related smart objects, it will be possible to view a deeper level of monitoring.

Thus, in the context of real time monitoring, five use cases were identified: View overall process status, View step status, View step details, View smart objects data and View legacy system data. Figure 91 shows real time monitoring use cases and its dependencies. Hereafter, the use cases are described and the related user interface mock-ups are shown.

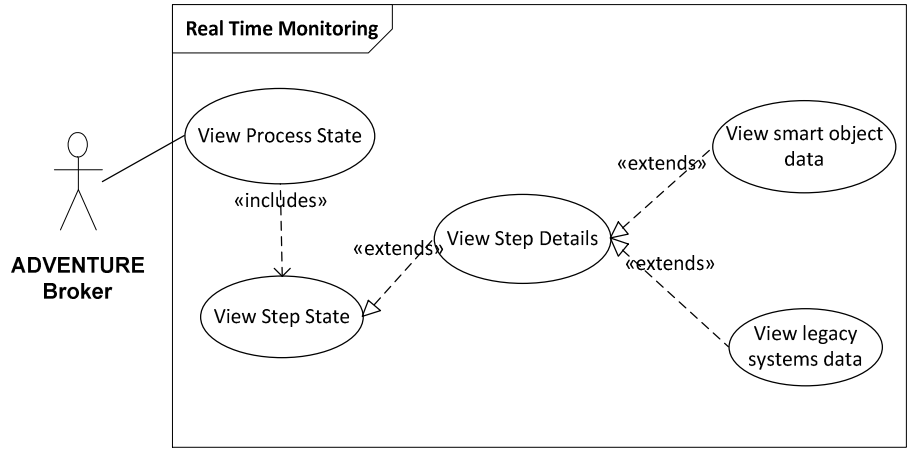

Figure 91 - Real Time Monitoring Use Cases

Use Case: PM-U01	Name: View Process Status
Description	Process status monitoring consists in a holistic view of the

	process status, including the status of each step. Users with the needed permissions will be able to see the steps started, running or terminated, on time or delayed or close to due date. The process model will be shown using interface as the process designer. On Mouse over tasks, some details about each task will be presented in a tooltip.
Actors	ADVENTURE Broker
Pre-conditions	• Process Owner (logged in with the right permissions to access process monitoring. • Smart Process fully configured and running in the Smart Process Engine.
Inputs	• Events from Process Execution Engine • Updated data regarding process execution, dashboard, legacy systems and smart objects.
Events sequence	• Broker drills down the active process list using the dashboard (Level 0) process menu • Broker clicks on a specific process • Broker clicks monitor process • The monitoring module user interface is shown and the process model is automatically shown with the current status of the process. • It will be possible to see the process path by clicking on “path” button. • Process and step properties can be seen in two different grids below and at the right of the process model.
Extension points	Use Case: PM-U02 “View Task Status”
Outputs	Holistic graphical and real time view of the process status.
Post-conditions	N/A
Requirements map	F30,F33,F37,F38,F59
Related Mock up	Figure 92 - Real-time Monitoring

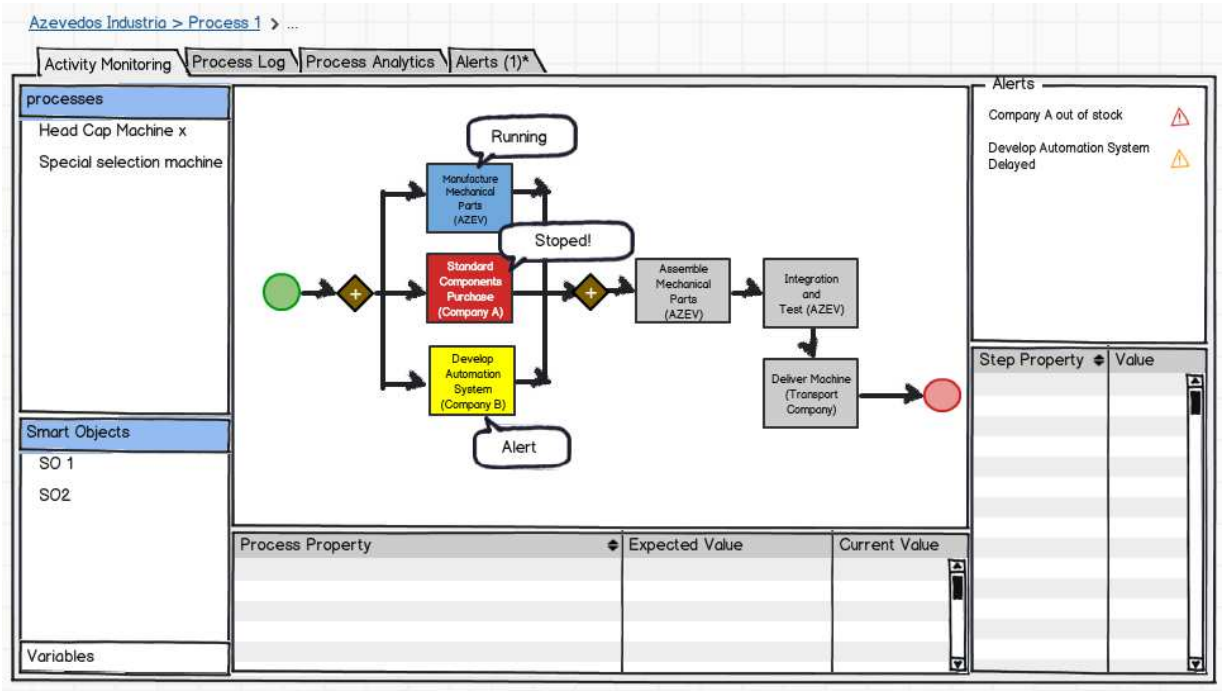

Figure 92 - Real-time Monitoring Panel Mockup

Use Case: PM-U02	**Name: View Task Status**
Description	The task status summary is presented in the global process monitoring view as shown in Figure 92, and on mouse over some more details about each task are presented by means of a tooltip, as shown in Figure 92.
Actors	ADVENTURE Broker
Pre-conditions	• Process Owner logged in with the right permissions to access process monitoring • Smart Process fully configured and running in the Smart Process Engine
Inputs	• Events from Process Execution Engine • Process Model constraints defined during the process design.
Events sequence	• Broker drills down the active process list using the dashboard (Level 0) process menu • Broker clicks on a specific process • Broker clicks monitor process • The monitoring module user interface is shown and the process model is automatically shown with the current status of the process • Process and step properties can be seen in two different grids below and at the right of the process model • Each of the steps presents a summary of its status
Extension points	Use Case: PM-U03 – “View Step Detail”

Outputs	Task specific details
Post-conditions	N/A
Requirements map	F30,F33,F37,F38,F59
Related Mock up	Figure 93 - Real-time Process Monitoring Panel

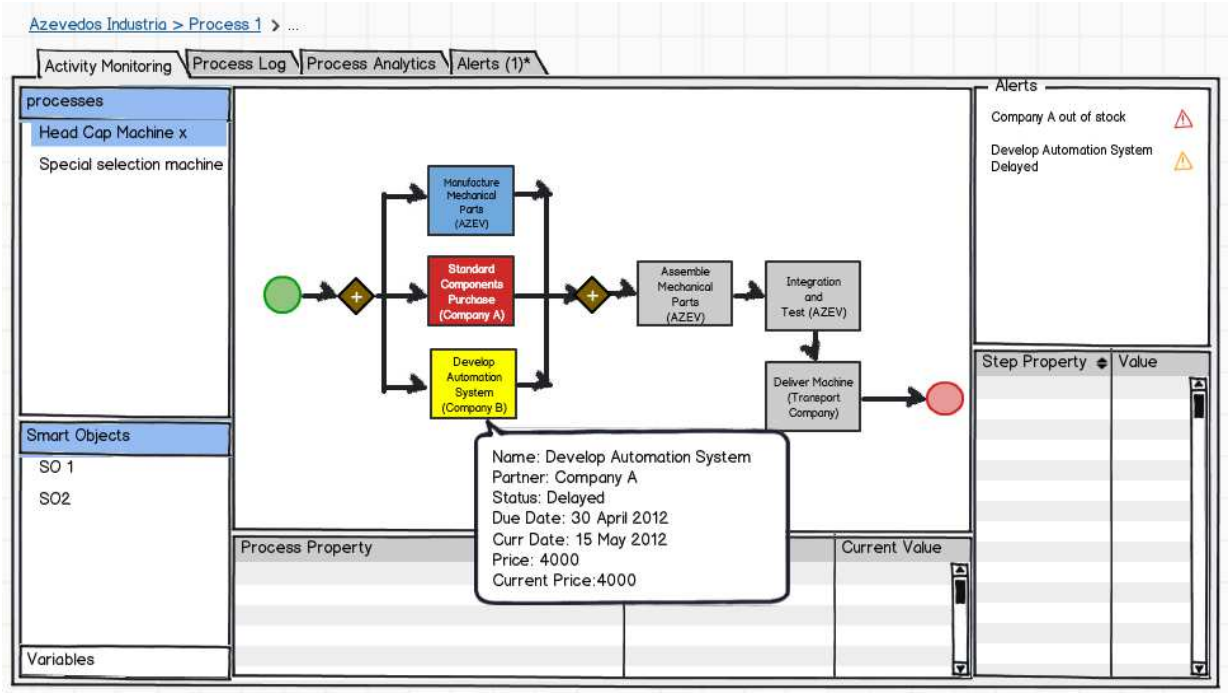

Figure 93 - Real-time Process Monitoring Panel Mockup

Use Case: PM-U03	**Name: View Step Detail**
Description	Besides the task status summary shown in use case **PM-U02** more detail can be visualized if the user clicks on a specific task, switching to the UI level 3 (task level). Here the specific details regarding services, assigned partners and task execution log can be visualized and updated.
Actors	ADVENTURE Broker, ADVENTURE Partner
Pre-conditions	• Smart Process Running on SPE and configured access rights.
Inputs	• Data from Legacy Systems and smart objects, data from process models and data introduced manually in the dashboard.
Events sequence	• The broker clicks on a specific task in the global process monitoring • Broker enters the task level monitoring page. • Specific information about the task and associated service

	is presented • The partner that is providing a service is able to update the task status by entering data and click update
Extension points	N/A
Outputs	Task details data.
Post-conditions	N/A
Requirements map	F30,F33,F37,F38,F59
Related Mock up	Figure 94 - Real-time Task Monitoring Panel

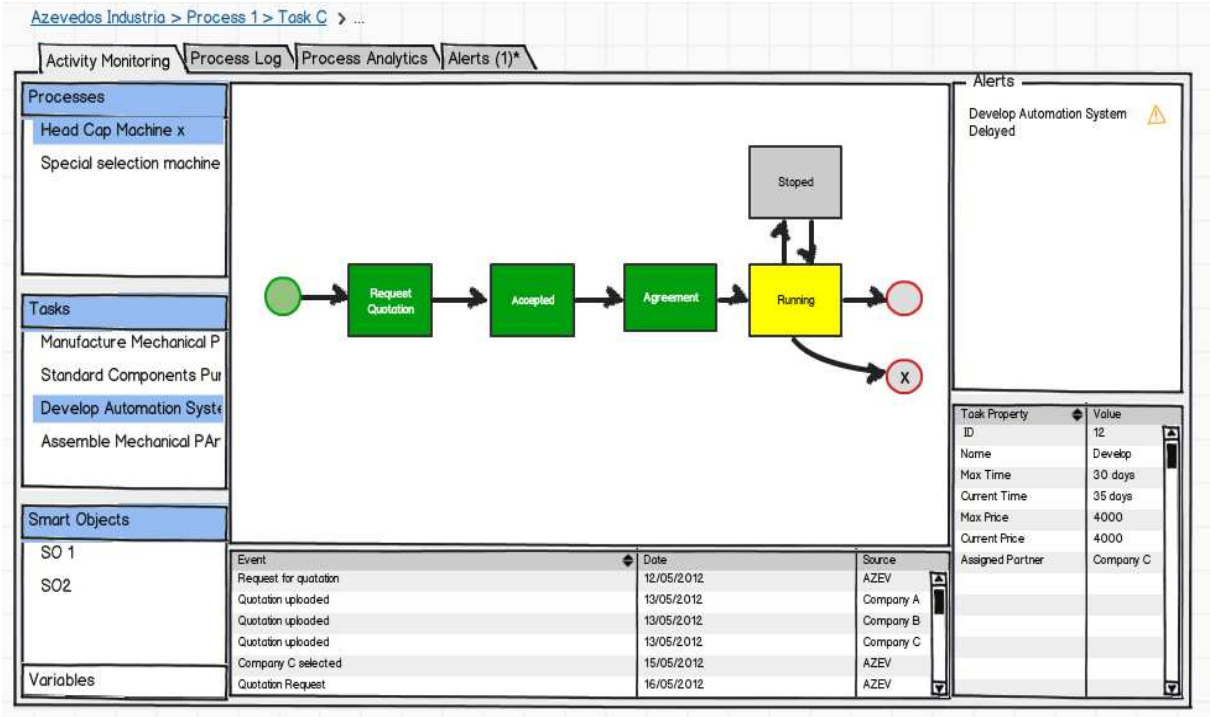

Figure 94 - Real-time Task Monitoring Panel Mockup

Use Case: PM-U04	**Name: View Smart Objects Data**
Description	Smart Objects are associated with tasks and processes. Thus it will be possible to view the list of smart objects in the scope of a process or a task. For instance, a standard part bought by AZEVEDOS is being transported by ship and has a wireless sensor, so it can be tracked. For instance, the location of the package, shock level, temperature will be available in the monitoring panel when the Smart object is selected.
Actors	ADVENTURE Broker, ADVENTURE Partner
Pre-conditions	• Smart Object configured and assigned to the task.
Inputs	• Data from the smart objects
Events sequence	• In the task monitoring page, user is able to view the list of smart objects that are related to the specific task

	• Broker clicks a specific smart object • The list of related variables is presented in a grid with the name and value. • A tracking option can be enabled for a given variable. • A smart object report with graphical analysis is then presented to the user.
Extension points	N/A
Outputs	Smart Objects data available in the context of a smart process /task
Post-conditions	N/A
Requirements map	F66, F73
Related Mock up	Figure 95 - Smart Objects Monitoring

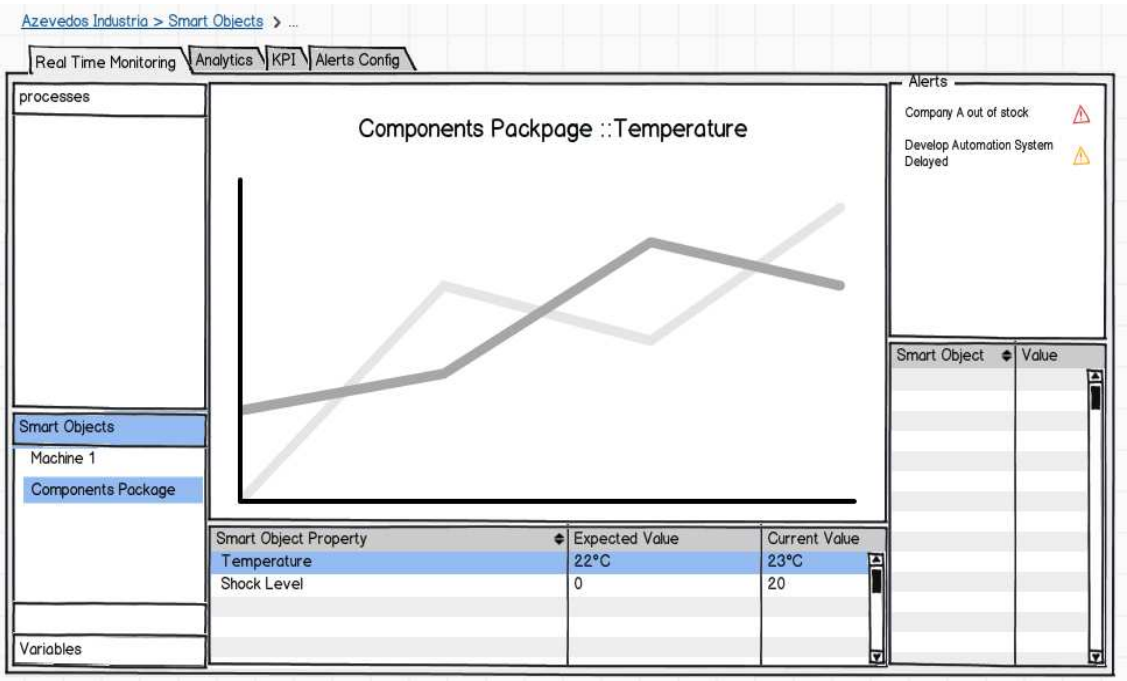

Figure 95 - Smart Objects Monitoring Panel Mockup

Use Case: PM-U05	**Name: View Legacy System Data**
Description	This use case refers to the data directly provided by partner's legacy systems such as supplier stock levels. A micro-flow process is used to get data from the legacy systems and show it in a graphical way using the monitoring panel. The process should be triggered by the process execution engine and should be performed with the help of a Gateway.
Actors	ADVENTURE Broker
Pre-conditions	• User with permission to access partner's data logged in the

	system.
Inputs	• Smart Objects Data
Events sequence	• Broker enters the real time monitoring page, • Broker selects Variables Menu and then chose a partner • Broker selects the available variables to track • A graphical display of those variables are shown.
Extension points	N/A
Outputs	Graphical view of the legacy system object variables
Post-conditions	N/A
Requirements map	F30
Related Mock up	Figure 96 - Legacy System Monitoring Panel

Figure 96 - Legacy System Monitoring Panel Mockup

5.11.2.2 Alerts and Notifications Use Cases

Process Monitoring will also include a rule engine that deals with creating notifications, alerts and messages that may:

- Throw alerts to the dashboard alert page
- Trigger the creation of new process instances
- Trigger the adaptation of a process, e.g. the automatic selection of a new virtual factory stakeholder, to improve delivery time
- Stop a process and require action from an ADVENTURE broker

These rules will be configurable via Application Configuration, however this functionality will be provided by the monitoring component.

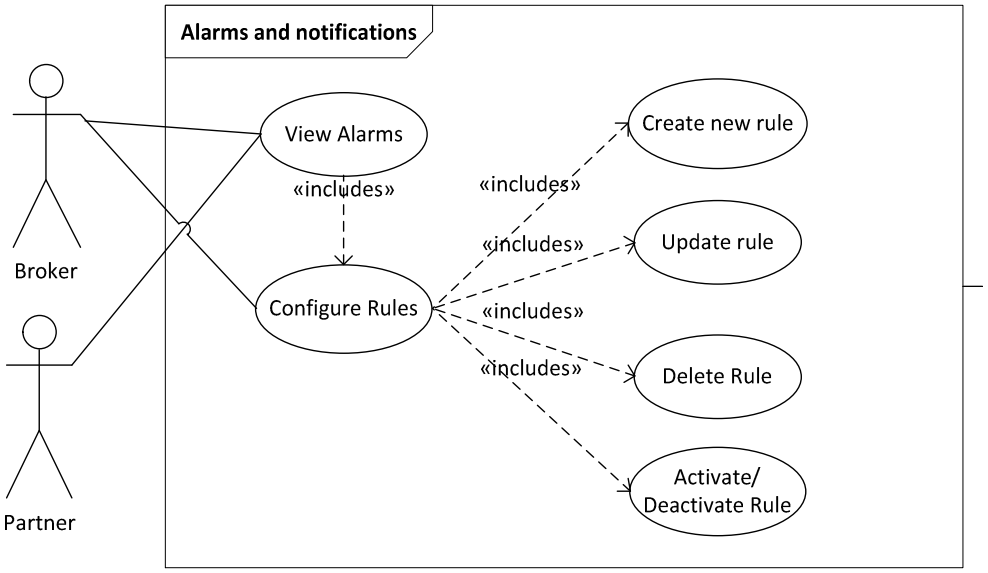

Figure 97 - Alarms Use Cases

Use Case: PM-U06	Name: View Alarms
Description	Alarms and notifications thrown by the Monitoring rules engine will be displayed in the alarms section.
Actors	ADVENTURE Broker, Partner
Pre-conditions	• ADVENTURE Broker or partner logged in the system
Inputs	• Alarms thrown by the Monitoring business engine
Events sequence	• When the rules engine triggers an alarm it is automatically shown in the alarms page. • It will be possible to “silent” the alarm for e.g. 10 seconds, delete the alarm or simply acknowledge it. • If the Broker silent the alarm it disappear for e.g. 10 seconds, after this time, if the alarm still active, it will appear again in the alerts page.

	• Acknowledge disables the alarm (grey colour) until the alarm condition disappears. • Delete simply deletes the alarm and it will never be triggered again.
Extension points	N/A
Outputs	Alarms are shown at the alarms page of the monitoring user interface
Post-conditions	N/A
Requirements map	F24, F33, F41
Related Mock up	Figure 93 - Real-time Process Monitoring Panel Figure 94 - Real-time Task Monitoring Panel Figure 95 - Smart Objects Monitoring Panel Mockup

Use Case: PM-U07	**Name: Configure Alarms**
Description	Although some of the alerts are automatic (e.g if task duration is greater than the desired duration entered during the process design, then an alarm will be triggered), user will be able to configure specific alarms related to process, task, smart objects or legacy systems variables through the rules configuration user interface.
Actors	ADVENTURE Broker, Partner
Pre-conditions	• ADVENTURE Member logged in the system
Inputs	• Events from Smart Process Engine

Events sequence	• Default:
	  • Broker Access Application Management Menu;
	  • Broker Clicks Configure Alarm Rules;
	  • Broker Clicks New Rule;
	  • Broker configures and saves rule
	• Using shortcut menu
	  • Using shortcut menu, user clicks configure over the object that he wants to monitor.
	  • Broker configures and saves rule.

Extension points	N/A
Outputs	Alarm Rule Configured and saved.
Post-conditions	N/A
Requirements map	F24
Related Mock up	Figure 98 - Monitoring Rules Configuration Panel and

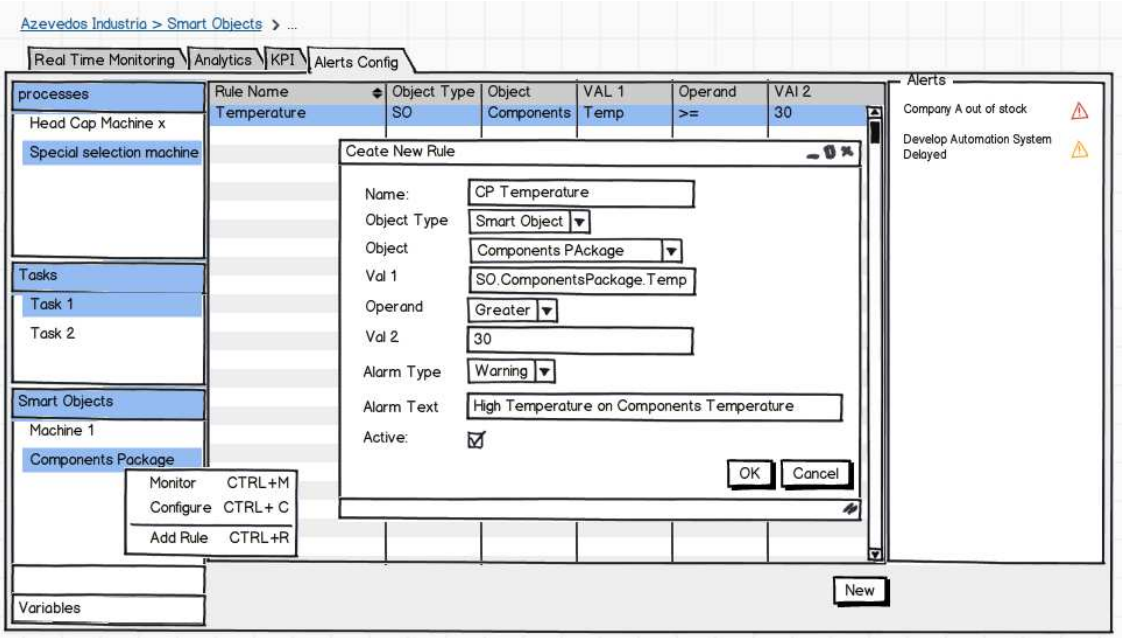

Figure 98 - Monitoring Rules Configuration Panel and Alarms

5.11.2.3 Process History Use Cases

In order to achieve efficient traceability, ADVENTURE will provide means to search for executed process instances. Relevant data collected during process execution will be stored at the cloud and thus available, later when a broker needs to access it.

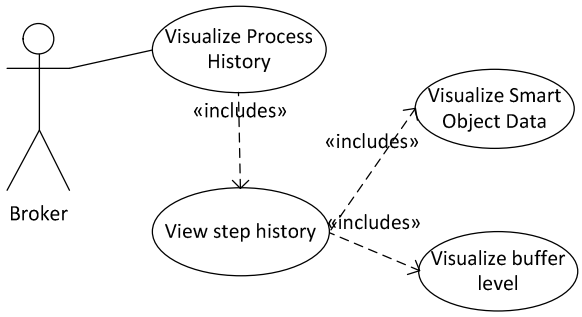

Figure 99 - Process History Use Cases

Use Case: PM-U08	Name: View Process History
Description	ADVENTURE process monitoring allows users to search for their own executed process instances and views its execution log. This functionality will be available at the process log tab in the monitoring user interface.

Actors	ADVENTURE Broker
Pre-conditions	• Broker logged in. Permissions to access process log.
Inputs	• Process Data stored during process execution
Events sequence	• Broker enters in the monitoring module • The Broker clicks in the process log tab and search for processes by name, process type, partners involved and date. • A process instance list matching the search is then presented • After clicking in a specific process instance, it will be possible to see the path, dates, partners involved, services involved, task details, etc. • All the information available in the real time monitoring will be displayed in the log.
Extension points	N/A
Outputs	Process instance log data stored in the cloud
Post-conditions	N/A
Requirements map	F30
Related Mock up	Figure 100 - Process Log

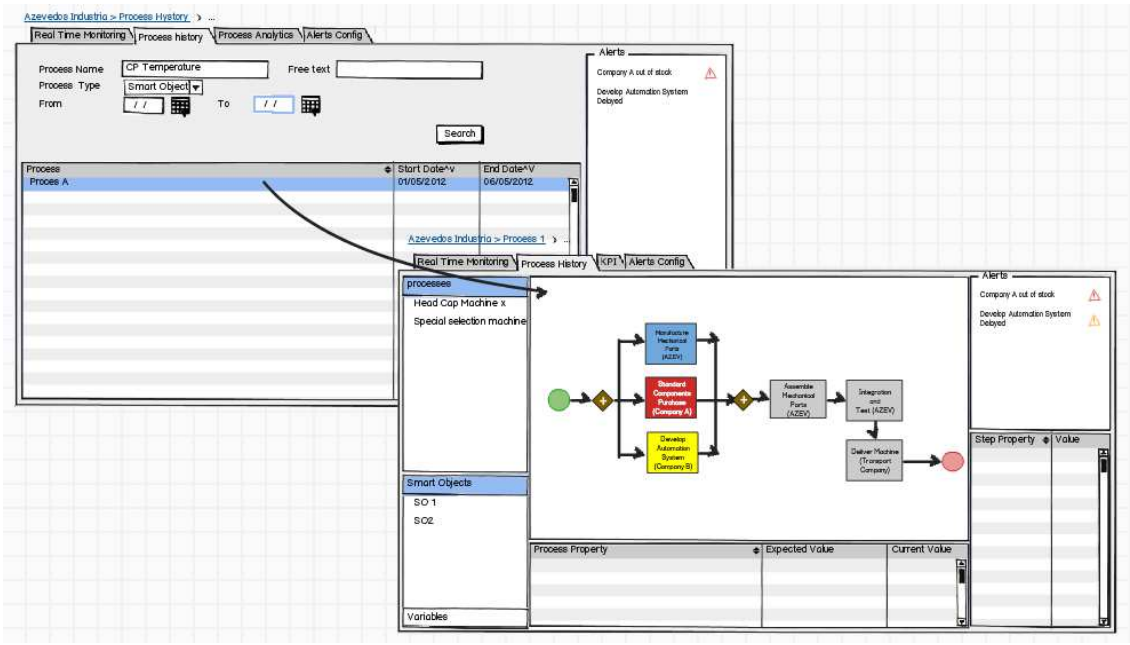

Figure 100 - Process Log

Use Case: PM-U09	Name: View Step History
Description	Besides the task status, detailed data can be visualized if the user clicks on a task, going to the level 3 (task level). Here the specific details regarding, services, assigned partners and task execution log can be visualized.
Actors	ADVENTURE Broker
Pre-conditions	• Process instance data stored in the cloud. User with permissions logged in the system.
Inputs	• Process instance data stored in the cloud
Events sequence	• Broker access menu process log and insert search criteria. • System shows the list of executed process instances matching the criteria. • Broker clicks one instance and the monitoring panel shows the path and all the related data that was stored during the execution.
Extension points	N/A
Outputs	• Process instance log data stored in the cloud
Post-conditions	N/A
Requirements map	F30
Related Mock up	Figure 100 - Process Log

5.11.2.4 Process Analytics Use Cases

The Process Monitoring component will provide means to evaluate the performance of processes and partners. Statistical data related to partner and activity performance will be considered in order to calculate configurable performance indicators. These performance indicators will help managers supporting their decisions.

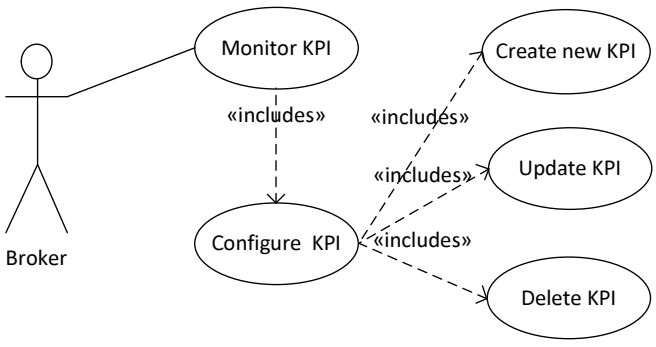

Figure 101 - Process Analytics Use Cases

Use Case: PM-U10	Name: Monitor Performance Indicator
Description	As a tool for continuous processes improvement, Monitoring will provide a set of indicators that show the performance that the different processes have had. Process Analytics presents

	summaries of the cases and activities that are currently closed. Task presents information of closed activities that belong to closed cases. Besides the default KPIs, it will be possible to define specific KPI and track them.
Actors	ADVENTURE Broker
Pre-conditions	• Broker logged in the system as ADVENTURE broker.
Inputs	• Process instance data stored during, performance indicators definitions
Events sequence	• Broker filters processes and selects KPIs he wants to analyze. • The system process all related data • The KPIs are presented graphically to the user so they can take decisions based on it.
Extension points	PM-U10 – "Define Performance Indicator"
Outputs	KPI Results are shown to the user in a graphic interface
Post-conditions	N/A
Requirements map	F32
Related Mock up	Figure 100 - Process Log

Use Case: PM-U11	**Name: Define Performance Indicator**
Description	As a tool for continuous processes improvement, Monitoring will provide a performance indicator editor which allows users to define performance indicators based on a set of metrics that are updated by the monitoring engine. Those performance indicators equations are then calculated by the process analytics component.
Actors	ADVENTURE Broker
Pre-conditions	• Broker logged in the system as ADVENTURE broker.
Inputs	• Process instance data
Events sequence	• Broker access Performance Indicators and clicks define new KPI • Broker selects the dataset that are needed to calculate the Performance indicator • Broker specifies the performance indicator equation • Broker saves the performance indicator definition.
Extension points	N/A
Outputs	Performance indicator definition is stored.
Post-conditions	N/A
Requirements map	F33
Related Mock up	N/A

5.11.3 Component Interaction

This section details the interaction between the Process Monitoring subcomponents with the other components of the ADVENTURE platform.

As stated in section 4, the Smart Process Engine (SPE) is the core component of this layer. The SPE orchestrates the invocation of services in the legacy systems of the virtual factory partners, as well as the gateway services that collect data from the smart objects and other external systems. The Monitoring Engine subscribes the events produced by the SPE and logs the relevant data in the cloud storage.

In order to maintain an updated data image of the virtual factory processes, the Monitoring Engine should capture the following events types from the SPE:

- Task is about to be executed
- Gateway service called
- Gateway service call finished
- Task is about to finish
- Task has failed
- Legacy System Data element was modified (E.g. stock level decreases by 10 units)
- Process model was changed
- Process model has an error
- Partner was changed
- Invalid Partner
- Process has been stopped / started / finished
- Process is stopping / finishing (intermediate state)
- Task execution progress (with task specific progress report format)

In run-time, the virtual factory broker may want to act upon a running process, e.g., stop or cancel the entire process, or just one of its particular steps. Since the SPE does not have its own user interface, the monitoring component captures such user commands and triggers the corresponding type of actions on the SPE:

- Halt or cancel the execution of a running process
- Restart the execution of a halted process
- Halt or cancel the execution of a particular step of a running process
- Restart the execution of a halted step
- Start a new process

Besides these actions, the Real-time Monitoring subcomponent will also forward notifications and alerts from the Rule Engine to the Dashboard.

The following Figure summarizes the main interactions of the Process monitoring component with the other components of the platform.

5.11.4 Conceptual Data Model

Two databases will be employed:

- Online Transaction Processing (OLTP) database for data entry and retrieval transaction processing where the system responds immediately to user requests.
- Online analytical processing, (OLAP) answering multi-dimensional analytical queries for business process management .

The following figure shows the main data entities that are manipulated by the process monitoring component.

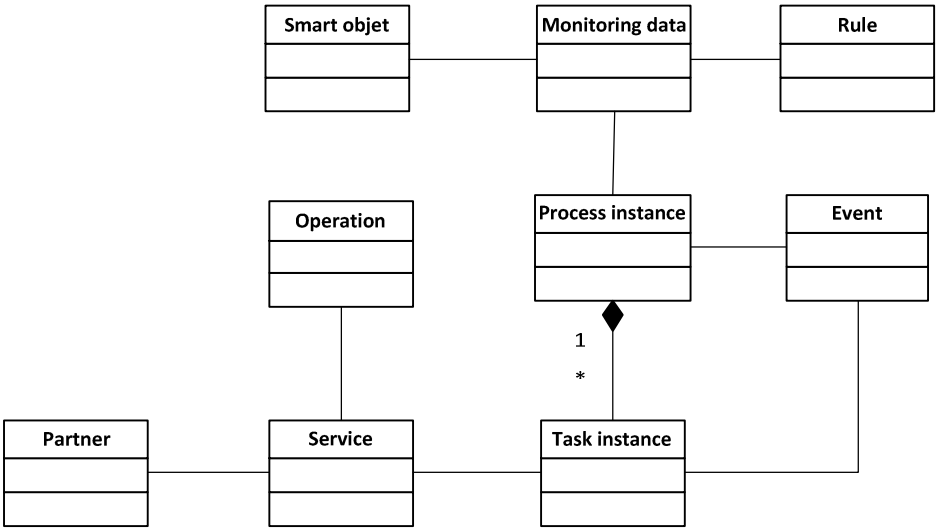

Figure 102 - Monitoring conceptual data model

5.11.5 Parameters To Take Into Account For Technical Specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+++
Regularly Updated	++
Technical Up-to-Datedness / Appeal	+++
Open Source	++
Non-Infecting	+++
Code-Quality	+++
Extensibility	++
Community	++
Performance	+++
Reuse of existing developments	+++
EU project origin	++
Platform (Portability)	+++
Open Standards Compliance	+++

Interoperability	+++
Specific Parameters	
User Interface aspect / quality	+++
Usability	+++
Modular	++
Web 2.0 Built-in graphic framework	+++
Quantity of graphic types	+++
External data loading	+++

5.12 Process Optimization

5.12.1 Overall Functional Characterization

The Optimization component provides a means to optimize Smart Processes regarding the assignment of partner services to the abstract process steps and activities, respectively, of a Smart Process.

As Smart Processes are initially modelled based on skill and technical requirements – utilizing the ADVENTURE Process Designer – rather than identifying one particular partner who offers a distinct service or pre-product, an abstract process model is used. This means that a step of the Smart Process is not directly assigned to a specific service of one partner, but it will be assigned to a set of potential partner services that can accomplish the task of this process step and realize this activity, respectively, according to their profile, i.e., their manufacturing capabilities and products, which have been defined and provided in the JOIN phase – utilizing the ADVENTURE Data Provisioning & Discovery component.

Having modelled an abstract Smart Process, an ADVENTURE Broker may assign (manually) certain (supplying) partner services or a group of partner services to the different steps of this process. But, the Broker may equally make use of the Optimization component instead. For this, it is required that either the Broker provides a set of potential partner services for each activity of the Smart Process, which are appropriate to realize those activities, or that the Optimization component takes care of this. The Broker and Optimization component, respectively, may use the ADVENTURE Data Provisioning & Discovery component to find such appropriate partner services for each activity as well as their properties regarding non-functional aspects as cost, delivery time, carbon footprint, etc. This information about non-functional properties is required to be able to distinguish partner services which are equally appropriate to realize an activity of a Smart Process. In fact, such information about non-functional properties may stem from two sources: (i) Simulations and historic process logs from monitored, past executions of single partner services; (ii) Static partner information which was entered by Active and Passive ADVENTURE Members during the JOIN phase.

Further, the Broker may also define requirements and constraints, respectively, and an objective for the overall Smart Process on those non-functional aspects, e.g., limitations in costs, delivery time, carbon footprint, etc., and provide this data to the Optimization component.

Having provided these input data, comprising the activities of the process, the process structure, and restrictions and constraints as well as an objective regarding non-functional aspects for the overall Smart Process, the Optimization component computes an optimized assignment of partner services to each activity of the Smart Process.

How this optimization is performed will be explained in the following. In order to calculate an optimized assignment of partner services to activities of the Smart Process, a mathematical optimization model is developed. For this, the provided structure of the Smart Process as well as the provided activities are recursively transformed into a proprietary tree based structure. This enables traversing the tree (and therewith the Smart Process) and connecting the provided candidate partner services with the respective activities. This way, it becomes possible to compute potential values for non-functional aspects (as, e.g., delivery time, costs, etc.) for the whole Smart Process.

In order to do this, the respective non-functional aspects' values of the candidate partner services are aggregated according to their position in the tree. This capability of computing aggregated non-functional values for the whole Smart Process is used to mathematically formulate and appropriately encode constraints on those non-functional values for the whole Smart Process. For this, the provided input data regarding restrictions on non-functional aspects (as, e.g., limitations in costs, delivery time, carbon footprint, etc.) is used. Analogously, the provided input data regarding the objective is used to mathematically formulate and appropriately encode objective functions (as, e.g., minimize total cost).

Having specified and encoded the optimization problem in an appropriate format (i.e., as a linear program) a standard linear programming solver, which is a standard software product for solving linear programs, can compute an optimized solution to this problem. As previously stated, the computation is thereby based on the set of potential partner services either provided by the Broker or identified and found by the Optimization component. Further, input data regarding the expected values for non-functional aspects may stem from past monitored executions as well as from performed simulations. The computation of a solution to the optimization problem is thereby strictly performed on a mathematical basis such that the provided restrictions and constraints are met and the objective is optimized. The solution is then transformed afterwards into an invocation plan, which constitutes the decision which partner services to assign to which activity of the Smart Process.

In order to clarify this functionality, an example is given. Assume the example process in Figure 103, which is written in Business Process Modelling Notation (BPMN).

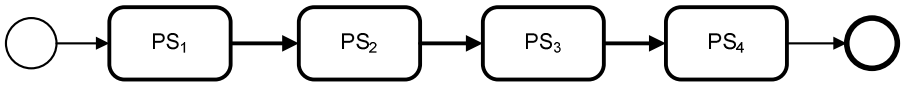

Figure 103 - Example Process

In this process, the activities for the process steps PS_1, PS_2, PS_3, and PS_4 are to be realized sequentially, i.e., one after the other. For PS_1, four partners offering respective manufacturing services S_{11}, S_{12}, S_{13}, and S_{14} have been identified by the Broker to be appropriate to realize PS_1. For PS_2, three services providers offering services S_{21}, S_{22}, and S_{23} were found by the Broker. For PS_3, again, four partners offering services S_{31}, S_{32}, S_{33}, and S_{34}, and for PS_4, three partners offering services S_{41}, S_{42}, and S_{43} have been investigated by the Broker. This situation, i.e., the example process as well as appropriate partner services are depicted in Figure 104. Generally, it has to be noted that it is not the purpose of the Optimization component to "find the best partner services" but to find and propose best partner services among available, functionally appropriate ones. Such available, functionally appropriate partner services are either provided directly by the Broker or found by using the ADVENTURE Data Provisioning & Discovery component.

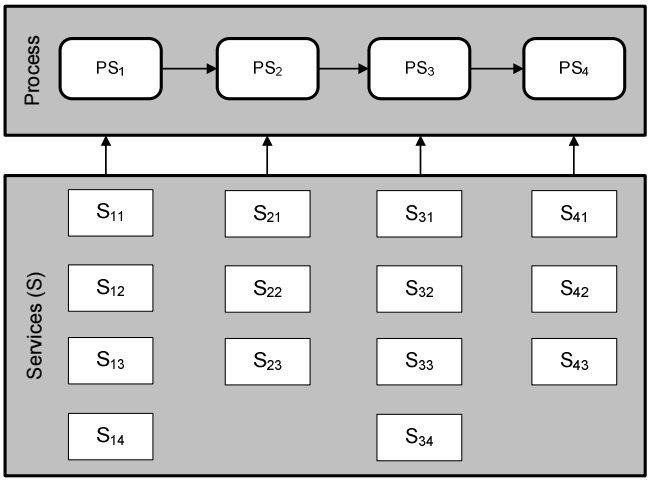

Figure 104 - Service Selection Problem

Regarding the example in Figure 104, the Broker may either manually assign partner services to the respective activities of the modelled Smart Process or may call the Optimization component, which computes and proposes an optimized invocation plan. In this example, the computation is thereby based on a restriction regarding the non-functional aspect “delivery date < 3 days” and on an objective to minimize total cost. An example for such an invocation plan, i.e., an assignment of partner services to activities, is shown in Figure 105. In this example, the partner services S_{11}, S_{22}, S_{34}, and S_{41} have been selected to realize the activities for the process steps PS_1, PS_2, PS_3, and PS_4.

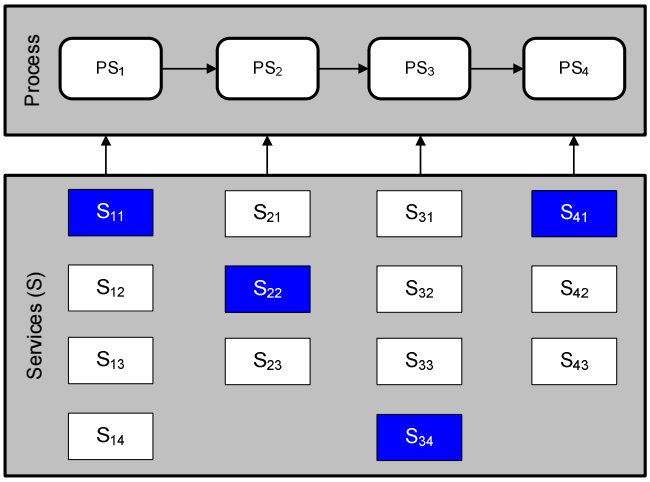

Figure 105 - Assignment of Services

For the sake of simplicity, an explicit assignment of one partner service to one activity of the process is assumed in this example. Actually, it could also be the case that individual services can realize more than one activity (which is not indicated in the above example but would be possible to be accounted for) or that the combination of two services, e.g., S_{11} and S_{22} are better than the combination of S_{12} and S_{23} even if the individual services S_{12} and S_{23} were better. This means that there can be dependencies between partners. For example

partner A may deliver one part in 5 days and partner B another part in 10 days but instead partner C and D may deliver each in 8 days meaning that still the combination C & D may be optimal.

For different subsets of partner services, different objectives, or different constraints, the Optimization component provides (most probably) different invocation plans. This assignment is provided to the Broker in a graphical way by the Optimisation component, which is indicated in Figure 106 (note that the rectangles represent the selected services). Thus, the Optimization component offers the Broker an alternative from manual selection on which concrete partner services to invoke for each of the activities. However, the final decision and actual selection of partner services is thereby left with the Broker.

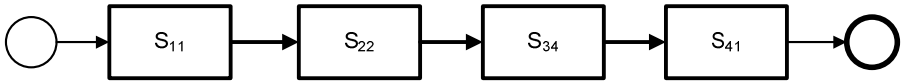

Figure 106 - Invocation plan

Also, in case of an event, which puts a successful execution of the designed Smart Process at risk or delays agreed delivery deadlines, the Optimization component may be utilized. For this, the “remainder” of the Smart Process, which has not been executed successfully as yet, is provided to the Optimization component. Also, an – under the new circumstances – updated set of appropriate partner services as well as adapted requirements and constraints can be entered into the Optimization component in order to retrieve a proposition, how to proceed the execution of the yet unexecuted part of the Smart Process, i.e., which partner services should be involved, in order to satisfy the constraints.

5.12.2 Use Cases

Referring to the requirement F79, it should be possible to provide the Optimization component with restrictions and objectives on non-functional properties for the (whole) Smart Process. This means that the Optimization component must offer and provide an interface to the Broker which enables the specification of such restrictions and objectives. As mentioned in Section 5.12.1, these restrictions constitute, e.g., limitations on costs, delivery time, carbon footprint, etc. Objectives on the other hand indicate which preferences in terms of “which aims” a Broker pursues regarding the optimization. For instance, a Broker may aim at selecting those partner services which offer and provide required manufacturing services cheapest, or which realize the activities of a Smart Process at minimum delivery time. In this context, it has to noted that restrictions and objectives can be set both for single process steps and activities, respectively, (and therewith for single partner services) and for the whole Smart Process. A potential GUI is shown in Figure 107.

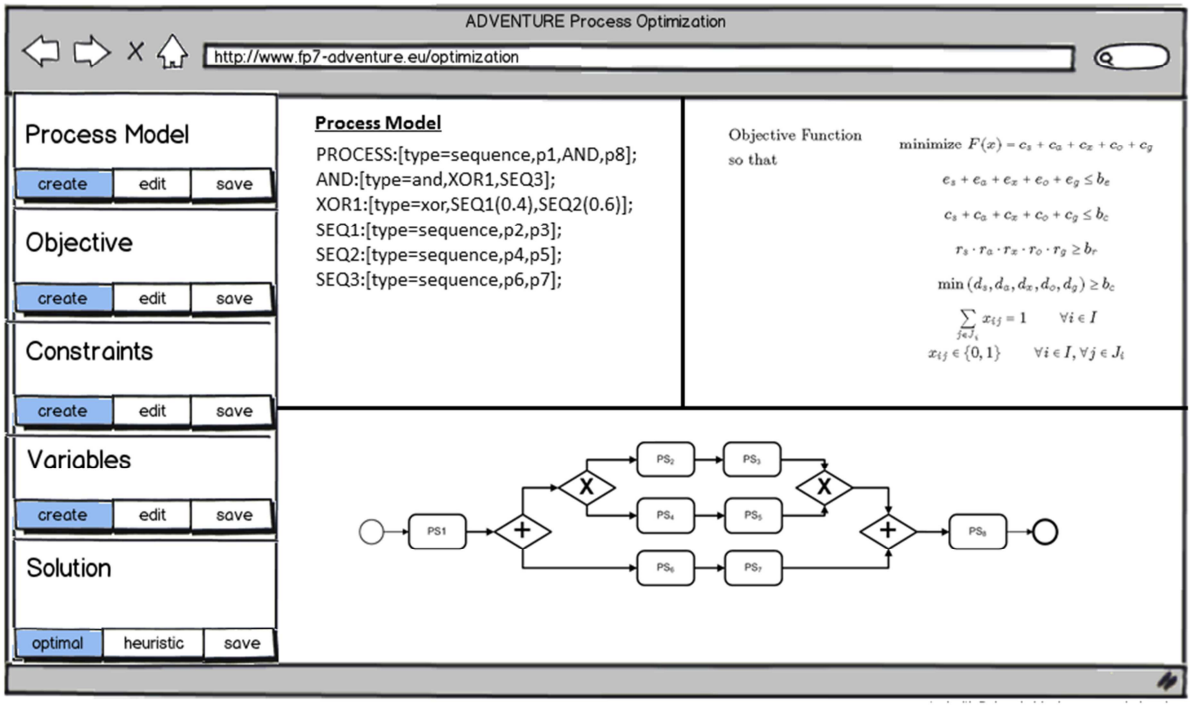

Figure 107 - GUI Optimization Component

Of course, this data can be provided automatically to the Optimization component from other ADVENTURE components, so that the Optimization component can act directly and automatically upon invocations from other ADVENTURE component.

Referring to the requirement F80, it should be possible to compute an optimized invocation plan. This means that the Optimization component must provide one or more suggestions about which partner services to select. This is based on the structure of the Smart Process, the subset of potential partner services appropriate to realize the activities of the Smart Process, and based on constraints as well as objectives on non-functional properties, provided by the Broker. In this context, it has to be noted that requirement F79 actually has to be satisfied in order to fulfil requirement F80. Thus, the following functionalities have been identified from the requirements:

- **Process description**
  - In order to optimize a process model, the respective model is encoded and provided to the Optimization component in an appropriate format. This will be performed by either automatically invoking component or manually by the Broker.
- **Assignment of candidate partner services**
  - The Broker has the possibility to explicitly provide the Optimization component with information about which potential partner services may come into consideration for which activities of Smart Process. If such an assignment is not preset manually, the Optimization component will trigger the ADVENTURE Data Provisioning & Discovery component to find such appropriate partner

services for each activity as well as their properties regarding non-functional aspects as cost, delivery time, carbon footprint, etc.

- **Develop an optimization model**
  - The Optimization component enables the Broker to manually specify an optimization model comprising an objective and constraints. Also, an automated generation of optimization models will be possible if the Optimization component is triggered by another ADVENTURE component. It will further be possible to provide multiple objectives, but they have to be prioritized by applying weights for all objectives.

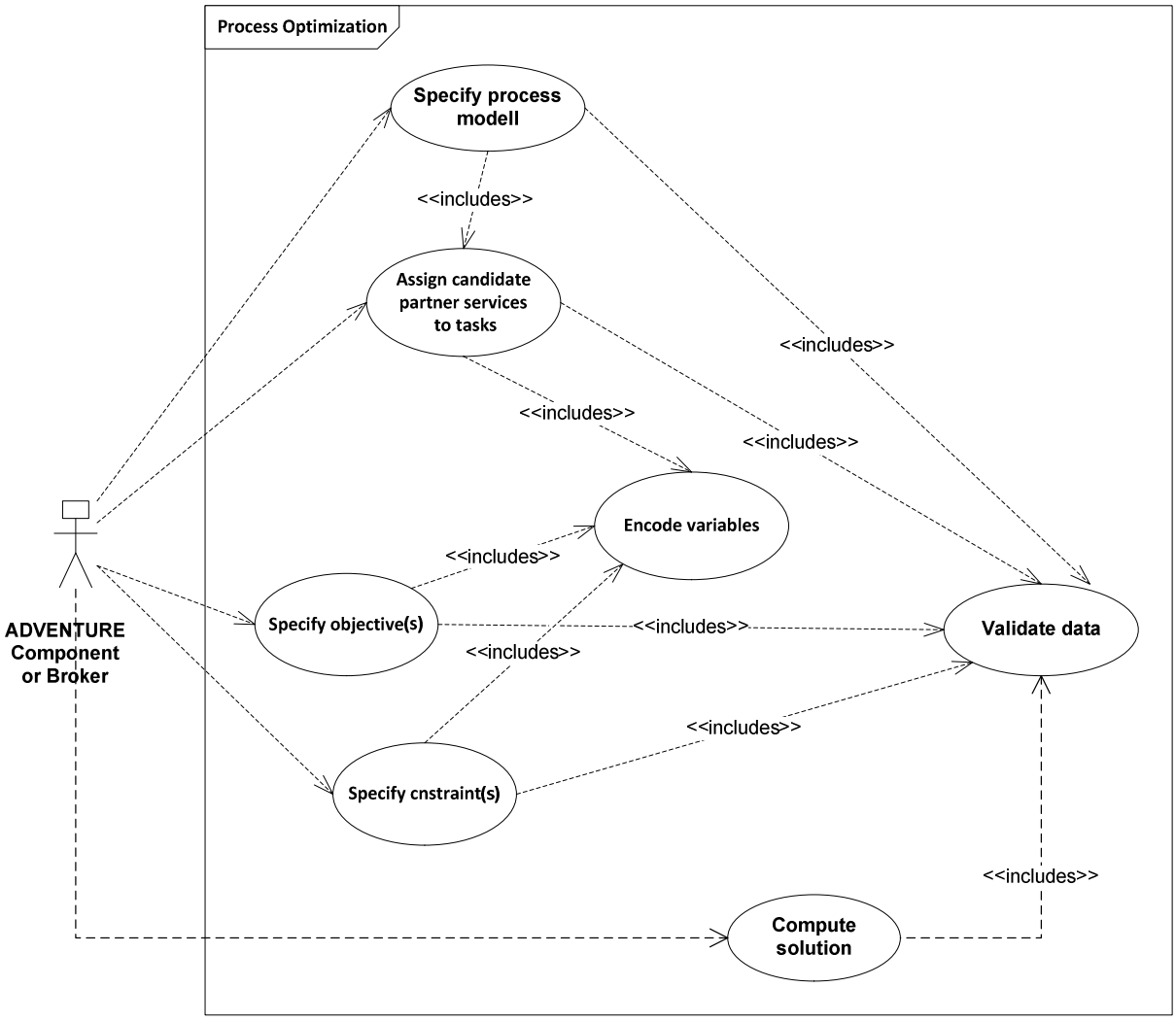

Figure 108 - Optimization Use Case Diagram

Use Case: PO-U01	Name: Specify process model
Description	The Optimization component requires the model of the Smart Process which is to be optimized to be encoded and provided to the optimization component in an appropriate format. This will be done either automatically by the invoking component or by manually by the Broker.
Actors	• ADVENTURE component • ADVENTURE Broker
Pre-conditions	• The process model has to be encoded in an appropriate format
Inputs	• Process structure and activities
Events sequence	• An ADVENTURE component or an ADVENTURE Broker encodes a new process model or imports/opens an existing one • The Optimization component creates respective variables representing the activities of the Smart Process and its structure
Extension points	• Use Case PD-U01 • Use Case PD-U03 • Use Case PD-U04 • Use Case PD-U05 • Use Case PD-U06 • Use Case PO-U03
Outputs	• Encoded process model
Post-conditions	N/A
Requirements map	F80
Related Mock up	Figure 107

Use Case: PO-U02	Name: Assign candidate partner services to activities
Description	To make sure that the Optimization component actually "can" compute an assignment of partner services to the activities of a Smart Process, potential candidate partner services have to be identified for each activity. This can be either done manually by a Broker according to his experience/preferences/requirements, automatically by the ADVENTURE component which triggers the Optimization component, or by the Optimization component itself. For this, the Optimization component uses the ADVENTURE Data Provisioning & Discovery component to find appropriate candidate partner services.
Actors	• ADVENTURE component • ADVENTURE Broker
Pre-conditions	• A process model has been created/ provided • At least one candidate partner service must be assigned to each activity of the process model
Inputs	• Activity of a process model

Events sequence	• An ADVENTURE component or ADVENTURE Broker opens/loads a process model • For each activity of the process model, potential candidate partner services are assigned • Respective non-functional properties for each candidate partner service is fetched using the Data Provisioning and Discovery component • The Optimization component creates respective variables representing the candidate services for the activities of the Smart Process and their respective non-functional properties
Extension points	• Use Case DD-U03 • Use Case DD-U04 • Use Case DD-U05 • Use Case PO-U03
Outputs	• Assignment of appropriate candidate partner services for a activity
Post-conditions	N/A
Requirements map	F80
Related Mock up	Figure 107

Use Case: PO-U03	**Name: Encode variables**
Description	In order to formally, mathematically specify objectives and constraints as well as the assignment of potential candidate partner services, variables are required to represent and encode such objectives, constraints, and candidate partner services. These variables are used to formulate and develop the optimization model. For instance, a bound that is required to create a constraint on, e.g., delivery time, is created and encoded with “b_e”.
Actors	Optimization component
Pre-conditions	• A process model has been created/ provided
Inputs	• Encoded process model • Candidate partner services • Constraint(s) • Objective(s)
Events sequence	• Specification of process models triggers the creation of variables • Specification of objective(s) triggers the creation of variables • Specification of constraints(s) triggers the creation of variables • Assignment of candidate partner services triggers the creation of variables
Extension points	N/A
Outputs	• Variables

Post-conditions	N/A
Requirements map	N/A
Related Mock up	Figure 107

Use Case: PO-U04	**Name: Specify objective(s)**
Description	The Optimization component requires to be provided with an objective which should be pursued. For instance, the total cost should be minimized.
Actors	• ADVENTURE component • ADVENTURE Broker
Pre-conditions	• A process model and candidate partner services have been created/provided
Inputs	• Target state, i.e., minimum/maximum
Events sequence	• An ADVENTURE component or ADVENTURE Broker specifies whether a minimization of maximization of a certain non-functional aspect should be achieved • The Optimization component uses the created, respective variables representing the candidate partner services and their non-functional properties and creates a mathematical expression indicating an objective on a non-functional service property
Outputs	• Objective(s)
Extension points	• Use Case PO-U03
Post-conditions	N/A
Requirements map	F79
Related Mock up	Figure 107

Use Case: PO-U05	**Name: Specify constraint(s)**
Description	The Optimization component requires to be provided with the constraints restricting the assignment of partner services such to satisfy lower/upper bounds. For instance, the total cost must be lower than 5,000 EUR.
Actors	• ADVENTURE component • ADVENTURE Broker
Pre-conditions	• A process model and candidate partner services have been created/provided
Inputs	• Lower/upper bounds of non-functional properties, e.g., cost
Events sequence	• An ADVENTURE component or ADVENTURE Broker specifies a constraint on a certain non-functional property • The Optimization component uses the created, respective variables representing the candidate partner services and their non-functional properties and creates a mathematical expression indicating a constraint on a non-functional service property
Outputs	• Constraint(s)
Extension points	• Use Case PO-U03

Post-conditions	N/A
Requirements map	F79
Related Mock up	Figure 107

Use Case: PO-U06	**Name: Compute solution**
Description	Computes an invocation plan, i.e., an assignment of partner services to the activities of a process model optimizing the objective and satisfying the constraints.
Actors	• ADVENTURE component • ADVENTURE Broker
Pre-conditions	• A process model, candidate partner services, objective(s), and constraints have been created/provided
Inputs	• Process model • Objective(s) • Constraint(s)
Events sequence	• The Optimization component develops an optimization problem by assembling a process model with respective candidate partner services, (a) constraint(s), and (an) objective(s) • An ADVENTURE component or ADVENTURE Broker computes a solution to the developed optimization problem
Outputs	• Invocation plan
Events sequence	N/A
Post-conditions	N/A
Requirements map	F80
Related Mock up	Figure 107

5.12.3 Component Interaction

Referring to the architectural specification in D3.1, the Optimization component is mainly triggered by the Process Designer, Forecasting & Simulation, or directly by the Broker but it may equally be invoked by any “other component” which provides appropriate input data (cf. Section 5.12.1) and requires process optimization. Its interaction with other components is visualized in Figure 109.

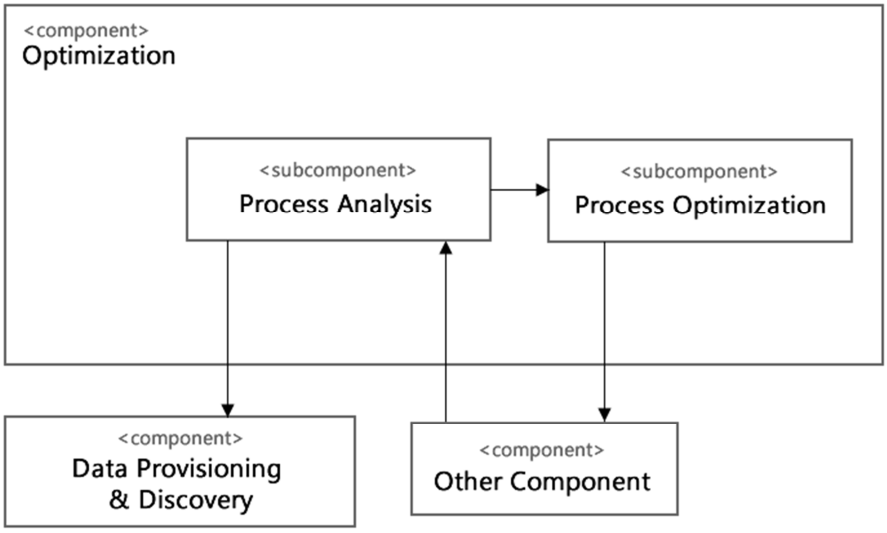

Figure 109 - Component Interaction

After being triggered and having received appropriate input data, the Optimization component analysis's this data in order to make sure that a valid process model as well as respective constraints and objectives have been provided. In fact, this is the duty of the Optimization's Process Analysis subcomponent.

After having approved the validity of the input data, the Optimization component interacts with the ADVENTURE Data Provisioning & Discovery component in order to either enable the Broker to define a set of potential partner services for each activity of a Smart Process or to fetch information about appropriate partner services on its own, utilizing the semantically enriched descriptions of required functionalities for each of the steps of the Smart Process. Respective data regarding the non-functional properties of the potential partner services is also collected via the Data Provisioning & Discovery component.

Having collected all the data necessary to perform the actual optimization, the "Process Optimization" subcomponent optimizes the Smart Process and provides the computed invocation plan back to the calling component.

5.12.4 Conceptual Data Model

The main data that has to be designed and exchanged with the Optimization component comprises information about the process model, which is to be optimized, i.e., the structure of the process and candidate partner services, and data about non-functional requirements and constraints as well as objectives.

The data format (conformant with that provided by the Process Designer) will be structured text, most likely in an XML format. In order to describe the structure of the process model, specific key terms will be used to indicate structural elements of the process (as, e.g., "AND", "XOR"). The data representing eligible candidate partner services will contain the partner names or other (but unique) identifiers and a reference to which steps of the provided process model the respective partner may be assigned. In addition, it will also contain non-

functional aspect(s) as well as corresponding quantitative value(s). These data will be fetched from and exchanged with the Data Provisioning & Discovery component.

For description of restrictions/conditions/objective(s) on non-functional properties for (whole) Smart Processes (cf. requirement F79), the respective data will contain the non-functional aspect(s), which are to be restricted (e.g., limitations in costs, delivery time, carbon footprint, etc.) or optimized (e.g., minimize total cost), as well as corresponding, quantitative upper/lower bounds. These data is to be specified and provided by the component that triggers the Optimization component or by the Broker itself.

The data representing the results of the Optimization component contain and describe an invocation plan. I.e., for each partner service, a quantitative value is provided which indicates the “suggestion” of the Optimization component to select this partner for a specific activity of the optimized process model.

5.12.5 Parameters to Take into Account for Technical Specification

Parameter	Importance (- - - +/- +++)
Generic	
Maturity & Stability	++
Regularly Updated	-
Technical Up-to-Datedness / Appeal	+++
Open Source	+
Non-Infecting	+
Code-Quality	-
Extensibility	-
Community	+
Performance	+++
Reuse of existing developments	++
EU project origin	+
Platform (Portability)	+
Open Standards Compliance	+
Interoperability	++
Specific	
Linear Programming Solver	+++
Efficiency	++
Scalability	+
Allow semantic annotation	++
Process model descriptions based on standard	+

5.13 Smart Object Integration

5.13.1 Overall Functional Characterization

Possessing a sufficient amount of context data is essential to manage processes of virtual factories efficiently and for example react to changing circumstances, e.g., in the context of a manufacturing process, as soon as possible. In ADVENTURE, an important data source for such context data will be Smart Objects which will, for example, provide real-time data from their environment to the ADVENTURE system and its users. This can for example be a machine which measures its temperature or a means of transport which measures shock and tilt values during transportation. To be of any use, this data has to be made available in the ADVENTURE system. Thus, a communication possibility between the ADVENTURE system and Smart Objects has to be established. In this context, different Smart Objects with different specifics, data sources, etc. have to be connected to the ADVENTURE system. This is the task for the Smart Object Integration (SOI) component in conjunction with appropriate gateways Thus, the SOI component takes care that bi-directional communication channels are established through Gateways between the ADVENTURE system and Smart Objects relevant for the ADVENTURE system so that gathered and relevant data can be accessed and transmitted from and to the ADVENTURE system and its users. Thus, from an overall point of view, the main task for the SOI will be to provide connection possibilities to different Smart Objects and using different specific Gateways which will provide the connection possibilities to ADVENTURE as access points to the system, so that a data exchange with the ADVENTURE system can take place.

5.13.2 Use Cases

Functional requirement F66 states that “Process monitoring should support Smart Objects with single or multiple sensors”. This implies that the SOI component has to offer the possibility to access data from a Smart Object, which possesses only one sensor, but simultaneously provide the access to different data from different sensors of a single Smart Object. Several modes of operation are possible in this context. Thus, a number of addressing options are available:

- A: Via a Smart Object: a data request to a certain Smart Object(s) would be directed and all data available from sensors of this Smart Object would be returned.
- B: Via a sensor of a Smart Object directly: a sensor(s) of a Smart Object can be individually queried for current data.
- C: Via the parameter to be measured by a sensor: all sensors from all Smart Objects matching a user query would return the current data.

Following an event-based push approach complementing the above described query-based addressing approach, similar functionality can be realized as follows:

- A: Based on a Smart Object: event data is pushed on Smart Object basis, meaning accumulated data from all sensors of a Smart Object is pushed.
- B: Via a sensor of a Smart Object directly: event data is pushed data on individual sensor basis, meaning all sensors individually push event data.

- C: Via the parameter to be measured by a sensor: all sensors that match a previously defined event description send data simultaneously.

Functional requirement F73 states that “Process monitoring data from Smart Objects is gathered”. This requirement implies that Smart Objects possess the ability, e.g., realized by being equipped with corresponding sensors, to gather data relevant for process monitoring. Additionally, it must be possible to at least buffer the gathered data locally on the Smart Object for a short time until the data is transmitted.

Functional requirement F74 states that “Process monitoring data from Smart Objects is locally pre-processed”. In order to save on transmissions, the processing capabilities of Smart Objects are exploited by pre-processing gathered data and for example dismissing irrelevant data or accumulating data as it is required. Consequently, different aggregation functions will be offered by Smart Objects, which will be individually callable. An example would be the average function, which would compute the average of different sensor readings from Smart Objects and would only transmit the average value and not each individual sensor reading.

Functional requirement F75 states that “Communication with Smart Objects, which monitor data which may be used in a process is enabled. They are connected to ADVENTURE through gateways”. This functional requirement indicates that it is not enough for Smart Objects to gather data, but simultaneously the data has to be made available to the system and its users. Thus, data transmission access and transmission possibilities will be provided via the SOI component in conjunction with relevant gateways. This allows the communication with Smart Objects in general and the collection of the monitor data gathered by Smart Objects (as sketched above) in particular.

Functional requirement F76 states that “Process monitoring data from Smart Objects is stored in ADVENTURE's data repository”. This requirement states the fact that, for example, for business process evaluation purposes, (historic) monitoring data gathered by Smart Objects must be made available to the ADVENTURE system and its users. It is not enough to provide only short-lived monitoring data in real time, but, e.g., a history of monitoring data must be kept as well. To ease the use of such historic data and allow simple and efficient access to it, the monitoring data gathered by Smart Objects should be made available in ADVENTURE’s data repository.

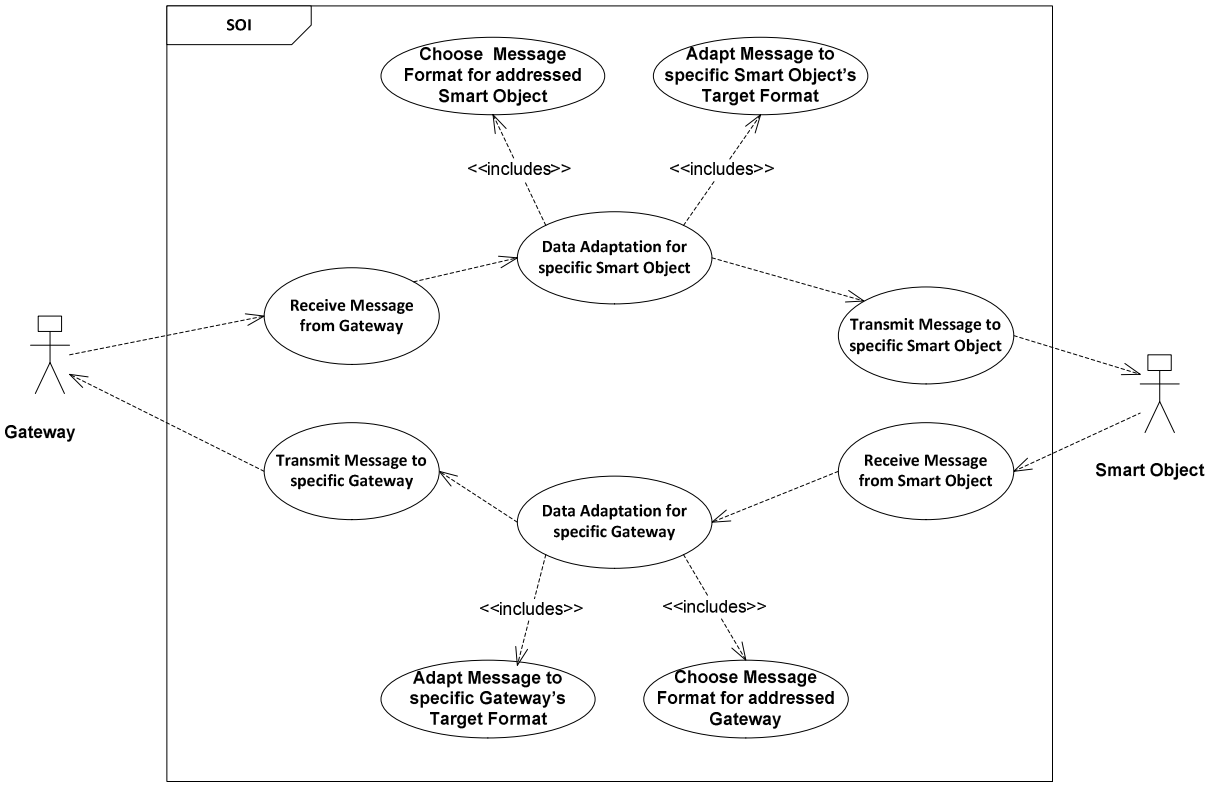

Figure 110 - Smart Object Integration Use Case Diagram

Use Case: SOI-U01	**Name: Receive message from gateway**
Description	The SOI component will receive messages to be forwarded to Smart Objects from Gateways
Actors	Gateway
Pre-conditions	• Communication connection to gateway exists
Inputs	• Message with receiver information
Events sequence	• Message channel established • Data transmitted
Extension Points	N/A
Outputs	• Received Message
Post-conditions	Message has been received correct and completely
Alternative flow	N/A
Requirements map	F66, F75

Use Case: SOI-U02	**Name: Receive message from smart object**
Description	The SOI component will receive messages from Smart Objects to be relayed to Gateways.
Actors	Smart Object

Pre-conditions	• Communication connection to Smart Object exists
Inputs	• Message with receiver information
Events sequence	• Message channel established • Data transmitted
Extension Points	N/A
Outputs	• Received Message
Post-conditions	Message has been received correct and completely
Alternative flow	N/A
Requirements map	F66, F75

Use Case: SOI-U03	**Name: Data adaptation for specific smart object**
Description	The SOI component has to take care that the message received from the Gateway and its information are transmitted to the receiving Smart Object in the expected and for the Smart Object processable data format and via the correct protocol. Thus, it has to adapt incoming messages to a format fitting to the receiving Smart Object.
Actors	SOI component
Pre-conditions	• Received message can be interpreted by the SOI component.
Inputs	• Received message
Events sequence	• Choose appropriate data format and protocol required by addressed Smart Object • Adapt message to target format
Extension Points	N/A
Outputs	• Adapted message or exception
Post-conditions	• Message fits specific Smart Object's data format and communication protocol.
Alternative flow	N/A
Requirements map	F66

Use Case: SOI-U04	**Name: Data adaptation for specific gateway**
Description	The SOI component has to take care that the message received from the Smart Object and its information are transmitted to the receiving Gateway in the expected and for the Gateway processable data format and via the correct protocol. Thus, it has to adapt incoming messages to a format fitting to the receiving Gateway.
Actors	SOI component
Pre-conditions	• Received message can be interpreted by the SOI component.
Inputs	• Received message
Events sequence	• Choose appropriate data format and protocol required by addressed Gateway • Adapt message to target format
Extension Points	N/A

Outputs	• Adapted message or exception
Post-conditions	• Message fits specific Gateways data format and communication protocol.
Alternative flow	N/A
Requirements map	F66

Use Case: SOI-U05	**Name: Choose message format for addressed smart object**
Description	The message format expected by the addressed Smart Object of the message received from the Gateway has to be identified.
Actors	• SOI component
Pre-conditions	• Information concerning the addressed Smart Object of a message to be forwarded to is available
Inputs	• Addressed Smart Object information
Events sequence	N/A
Outputs	• Expected target message format message has to be adapted to or exception
Post-conditions	N/A
Requirements map	F66

Use Case: SOI-U06	**Name: Choose message format for addressed gateway**
Description	The message format expected by the addressed Gateway of the message received from the Smart Object has to be identified.
Actors	• SOI component
Pre-conditions	• Information concerning the addressed Gateway of a message to be forwarded to is available
Inputs	• Addressed Gateway information
Events sequence	N/A
Outputs	• Expected target message format, message has to be adapted to or exception
Post-conditions	N/A
Requirements map	F66

Use Case: SOI-U07	**Name: Adapt message to specific smart object's target format**
Description	The message received from the Gateway is adapted to fit the target data format expected by the message's intended receiving Smart Object.
Actors	• SOI component
Pre-conditions	• Target format information is available
Inputs	• Received message • Target format information

Events sequence	N/A
Outputs	• Message adapted to target format or exception
Post-conditions	N/A
Requirements map	F66

Use Case: SOI-U08	**Name: Adapt message to specific gateway's target format**
Description	The message received from the Smart Object is adapted to fit the target data format expected by the message's intended receiving Gateway.
Actors	• SOI component
Pre-conditions	• Target format information is available
Inputs	• Received message • Target format information
Events sequence	N/A
Outputs	• Message adapted to target format or exception
Post-conditions	N/A
Requirements map	F66

Use Case: SOI-U09	**Name: Transmit message to specific smart object**
Description	A message in the format expected by its receiving Smart Object is transmitted.
Actors	• SOI component
Pre-conditions	• Message in target format for specific Smart Object is available
Inputs	• Message in target format for specific Smart Object
Events sequence	N/A
Outputs	N/A
Post-conditions	Message in target format has been delivered to specific Smart Object or exception has been raised
Requirements map	F66, F75

Use Case: SOI-U10	**Name: Transmit message to specific gateway**
Description	A message in the format expected by its receiving Gateway is transmitted.
Actors	• SOI component
Pre-conditions	• Message in target format for specific Gateway is available
Inputs	• Message in target format for specific Gateway
Events sequence	N/A
Outputs	N/A
Post-conditions	Message in target format has been delivered to specific Gateway or exception has been raised
Requirements map	F66, F75

5.13.3 Component interaction

The SOI component interacts via Gateways with other ADVENTURE components as depicted in Figure 111 which is based on the architecture specification in deliverable D3.1. Based on the type and capabilities of a Smart Object, the SOI component's functionalities, will be realized centrally on the Smart Object itself or might need to be realized in a distributed manner. Depending on the Smart Object itself, simply denoted as 'Device' in Figure 111, and its capabilities, e.g., processing power, energy supply, or memory size, several, or even all, functionalities will be provided by the Smart Object itself. The connector's functionality of providing a connection between Smart Object and the ADVENTURE system, e.g., via a wireless communication channel, will be provided by the Smart Object itself. The protocol conversion required to bridge between local communication with Smart Objects (e.g., via Bluetooth or ZigBee) and the ADVENTURE system, can be provided by a Smart Object in case it is a rather powerful device. However, it might need to be outsourced in case the Smart Object possesses only limited hardware. However, the protocol conversion within the integration engine will be provided in a generic way and instantiated on a case-specific basis by the connector subcomponent.

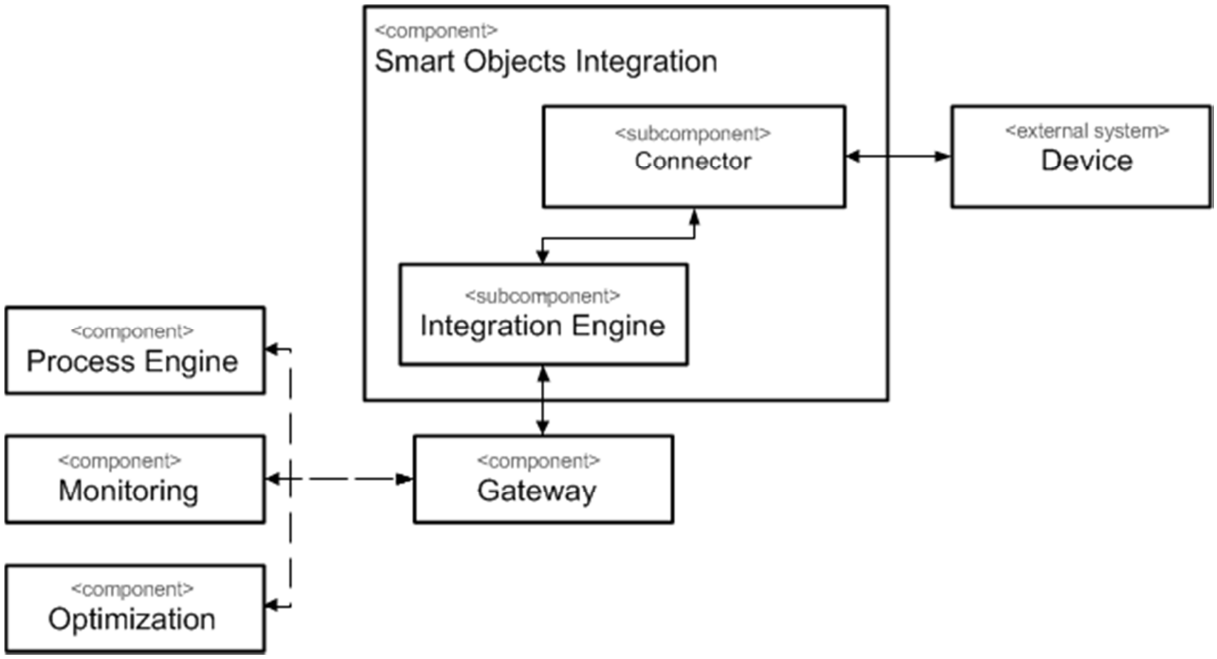

Figure 111 - The Smart Object Integration Component in the Architectural View

It is expected that Smart Objects will provide major input to ADVENTURE's process engine, ADVENTURE's monitoring component, and ADVENTURE's optimization component. Thus, the SOI component will mostly be used by these components and will provide data from Smart Objects to these components.

Two ways of interaction and data exchange are planned:

- A request/response-style (pull-based) interaction in which data is requested from ADVENTURE components like the process engine or the monitoring component. Here, the Smart Object device might offer data like a Web Service, which is offered and invoked via an appropriate gateway and the SOI component.
- A push-based interaction in which data is pushed from a Smart Object device via the SOI component and the gateway to 'interested' ADVENTURE components, like the process engine or the monitoring component.

As an example for the pull-based request/response-style interaction, one can imagine that the monitoring component currently needs an actual on-site temperature and calls the associated Web Service, which then provides the data from the related Smart Object device via SOI component and gateway. Concerning the push-based interaction, an example is where regular on-site temperature updates are required by an ADVENTURE component. In this case, the Smart Object would push data periodically to the ADVENTURE system via SOI component and gateway according to a locally specified transmission interval on the Smart Object device itself.

Additionally, push-based interaction might as well be realized according to the event-based communication paradigm. Here, certain 'normal' conditions could be saved on a Smart Object device and the device regularly checks current values against these conditions, e.g., specified in the form of thresholds, and only in case a threshold is violated, meaning an event occurred, it pushes data using the SOI component and gateway to the ADVENTURE system. The specific connection between Smart Objects and process instances will be realized on information basis, employing identifiers for the information provided by the Smart Objects and required by the process instance.

5.13.4 Conceptual data Model

As the SOI component will provide the communication from different specific Smart Objects to the ADVENTURE system and its components and vice versa through Gateways and thus will take care of transferring data gathered by Smart Objects, such data will constitute the major part which will be provided and manipulated by this component. The specific type of data gathered by Smart Objects and provided via the SOI component is strongly context-dependent. For example, if environmental temperature is a critical parameter to be monitored, Smart Objects, or more precisely sensors of the Smart Objects, will provide temperature measurement data. This data will be complemented with additional data, e.g., identifying the Smart Object the data belongs to or the sensor which measured the data. Additionally, it is imaginable that vibrations or humidity are relevant parameters. In this context, vibration and humidity data from Smart Objects and their sensors would be made available via the SOI component as described above.

5.13.5 Parameters to take into account for technical specification

Parameter	Importance (- - - +/- +++)
Generic Parameters	
Maturity & Stability	+
Regularly Updated	-
Technical Up-to-Dateness / Appeal	++
Open Source	-
Non-Infecting	-
Code-Quality	+/-
Extensibility	+/-
Community	+
Performance	++
Reuse of existing developments	+/-
EU project origin	-
Platform (Portability)	++
Open Standards Compliance	+/-
Interoperability	++
Specific	
Energy efficiency	+++
Cost efficiency	+
Wireless standard support	+
Legal usage in all EU states	+++
Extensibility	++
Ease of use	+
Support for variety of Smart Objects	+/-

6 Conclusion

This deliverable defines the functional specification of the ADVENTURE platform at two levels: overall system level and detailed component level. It is closely aligned with the previous deliverables of the project, particularly the user requirements report and the global architecture definition. In fact, the functional specification shows how the components of the architecture cooperate and it also contains the detailed functional specification of each component.

Starting from the system overview and user requirements, functionalities were identified and described. A conceptual arrangement of the ADVENTURE Dashboard was developed. It includes the identification of the common functionalities that will be provided by the platform as well as the positioning of the component specific user interfaces. Thus, a comprehensive vision of the ADVENTURE user interface was achieved.

In order to understand how the various components will cooperate in order to provide the desired functionalities, two use case scenarios were presented: one relating to process design and another one relating to process execution. These scenarios highlight how the system will work as a whole and enhance the global understanding of the system. Furthermore the scenarios will be employed during testing phases to verify the solution that will be developed.

For each one of the components defined in the global architecture, a detailed functional analysis was performed. An overall functional characterization of each component explains the main ideas on what the component will do and how. Also, for each component, the corresponding use cases were identified and described. The results of this functional analysis are an important input for software development tasks that will take place in work packages 4, 5 and 6. The interactions between the components were analyzed in detail and the services that each component will provide and consume were specified. Also a preliminary data model was developed for each one of the components. Finally, for each of the components, there's a table containing generic and component specific criteria which will be acting as an input to T3.3 for selecting technologies. The tables show how important a specific aspect is for a component (e.g. the performance).

As presented in section 5.1, the ADVENTURE Dashboard will be organized in four areas: Process Design, Process Management, Partner Profile and System Application. It also provides a secured session management system. The Dashboard will also provide a common framework for the implementation of the graphical interface.

The Process Designer component is the key tool in the stage of designing Virtual Factories. This component interacts with other supporting components such the simulation and optimization, to deliver ADVENTURE brokers with the core functionality to accomplish the Virtual Factory design process.

The Cloud Storage component will serve as the central data storage of the platform. It will allow the components to store different types of data (binary, structured, semi-structured and semantic) and it will ensure the high scalability required by the process related data.

The Data Provisioning and Discovery component provides a model and tools for structuring, publication and discovery of the key data of the ADVENTURE processes and partners, required for the design, management and operation of Virtual Factories.

The Message Routing component is located in the Data Exchange layer of the system architecture and will manage the secure exchange of messages and files between components and users/components.

The Transformation Service in the ADVENTURE Platform is responsible for transforming information between different data content formats. It runs in the data exchange layer, linked to the message routing system.

ADVENTURE Gateways will provide a framework for building custom made connections to external systems. These systems could either be an ERP system, other legacy systems, Smart Object, etc. The gateways will provide means to allow the exchange of information between the core components of the platform and the external systems.

The Smart Process Execution component will be at the heart of the ADVENTURE platform, as it will orchestrate all interaction in a virtual factory. Its purpose is to execute Smart Processes that were modelled in the Process Designer.

The Adaptation Component will provide an independent service with the purpose to handle all runtime aspects regarding the adaptation of process models.

The Forecasting & Simulation component will offer an independent service that relies on the Process Execution component and uses gateways to get temporal and execution information from partners without e.g. ordering something or invoking partner processes.

The Monitoring component provides the real time monitoring of ongoing processes, historical data relating to finished instances and business analytics relating to process and activities types.

The Optimization component provides a means to optimize Smart Processes regarding the assignment of partner services to the abstract process steps and activities, respectively, of a Smart Process.

In spite of the progress achieved during the functional specification task, the technical analysis of the system should be further developed during Task 3.3 – Technical Specification.

Annex A

This annex contains for each component of the system two tables showing the mapping between the functionality offered by the component and the requirements identified in D2.3 (requirements Analysis Report) that apply to the component.

Table 2 - Dashboard: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F14	1	Show if a potential change in your process (inside and outside of ADVENTURE) affects the client process.	Adapt process change
F31	1	Changing suppliers should be possible during process execution, in case of any the delivery risk. This implies the possibility to re-optimize the process. (see also F41).	Adapt supplier change
F41	1	The system must issue alerts (alert page) in case of unexpected situations that can arise during the production process: delays in tasks and their reasons (problem on a partner machine, shipping delays, goods damage, etc.).	Display alerts
F10	1	Create a browsable catalogue of ADVENTURE partners.	Display partners list
F70	1	Supplier side: Describe/specify offered manufacturing capabilities as services utilizing semantic annotation.	Visualize supplier's capabilities
F71	1	Customer side: Describe/specify required manufacturing capabilities as services utilizing semantic annotation.	Visualize customer's capabilities
F3	1	Allow Customers to add/enter data required by the PLUG process through the dashboard.	Update customer's profile
F1	1	During the PLUG process, partner information needs to be available through the dashboard. This can be static information, e.g., profile information, and dynamic information, e.g., current capacities.	Visualize partner's profile
F4	1	Allow Customers to monitor the progress of the PLUG process in the dashboard.	Visualize process monitoring
F30	1	The broker should be able to see on the Dashboard the stock levels at own factory, in transit, and at suppliers for any material.	Visualize updated stock
F37	1	The system must allow monitoring of process steps, providing status information of the activities in each of the partners (not started, late in x hours, in production, in testing, in packaging, in shipping, finished, % conducted) via the Dashboard. This information would come from F36 but maybe a subset of information (e.g. not all events - only important ones - e.g. started, errors etc.).	Display process status
F38	1	The system shall present the status of all process steps in a graphic manner. (e.g., by colour in each task	Display process update

		in the process, including status information and notes related to it). This builds of F37 to provide further UI improvements.	
F80	1	Show/propose optimized composition of services for a specific smart process.	Visualize optimized process
F28	1	Enable the broker to compute ATP ("available to promise") or CTP ("capable to promise") based on available resources, i.e., availability of material and capacity within the factory and also the availability of materials from suppliers.	Visualize partner's capacity and capability
F29	1	Enable the broker to gather dynamic, real-time information about, e.g., material stock position, capacity position, etc. (see also F28: this information will support the computation of ATP/CTP).	Visualize partner's capacity and capability dynamically
F2	1	The Plug process is automatically started whenever a broker adds a new partner.	Visualize new partner
F13	1	Show the suppliers (service providers) that have completed the PLUG process and are ready for selection by a broker. (see also F5 and Mock-up L1-3).	Visualize partner search
F77	1	Show services appropriate for realizing certain process steps. Initiated manually to query the ADVENTURE system.	Visualize process design
F54	2	The documents connected with process steps (or possibly the steps themselves) in the process step library should have access rights.	Visualize required documents
F65	2	Configurable dashboard - it should be possible to manage the ADVENTURE modules like components in iGoogle, for instance.	Configure dashboard
F59	2	All active processes should be viewable in the Dashboard.	Visualize all process overview
F33	2	The system must issue alerts (alert page) in case stock levels cross agreed thresholds (relates to F41).	Visualize process alerts
F63	2	Within the Process Execution Engine, if there are rules specified from a designed process which are broken, then an alert must be generated.	Visualize process alerts
F44	2	At the end of a simulation a simulation report has to be displayed and saved. This should include details about the configuration parameters, goals etc.	Visualize simulated result
F57	2	The system should provide descriptions, photos, catalogues, attached to every partner. These documents are sent when a potential client asks for details about their activities, capabilities.	Visualize required documents
F21	2	Clear visual representation of historic data.	Clear historic data
F22	2	Explore and display data granularity through dashboard.	Visualize historic data
F9	3	Allow Customers to express interest in the company's products (for further contact). (relates to F10).	Visualize partner's catalogue
F32	3	Notify partners about intended orders and processes with estimated definitions of product type, quantity and	Display notifications to partners

		delivery date.	
F85	3	Implement instant messaging for partners collaborating in a virtual factory integrated into the ADVENTURE system.	Visualize instant message
F40	3	View the sales trends and equipment investment trends for a set of selected companies. (see also F39).	Display forecasted results
F23	3	Presentation of interactive graphs / tables for data visualization is needed.	Visualize necessary data
F64	4	ADVENTURE Dashboard, or parts of it, should be available in a mobile environment.	Enable data visualization in mobile devices

Table 3 - Dashboard: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Visualize the adoption	Display potential change Display alternative partners Display alert page
1	Visualize the partners search	Display the search results Display the search results in detail Display the status of suppliers
2	Visualize the partner management	Display a table to specify the required manufacturing capabilities Display the partner information
2	Visualize the partner's profile	Display a table to add data of partner
1	Visualize the process monitoring	Display the progress Display the stock level Display the process steps Display the status information of activities Display real-time information Display active processes Display alert page
1	Visualize the optimization	Display the optimized composition of services
1	Visualize the simulation	Display the simulation report
1	Visualize the forecasting	Display forecasting result
1	Visualize the process design	Display alternative/recommended partners
1	Visualize the process search	Display the search field Display comparison report
3	Visualize the documents management	Display the access right of documents
2	Visualize the message exchange between users	Display message to the user
2	Visualize the configuration	Display configurable dashboard

Table 4 - Process Designer: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F3	1	Allow Customers to add/enter data required by the PLUG process through the dashboard.	Support modeling of human tasks in process models
F5	2	Define the PLUG process, i.e., prerequisites of a supplier, in order to enable a broker initiating collaboration with this supplier.	Design process models
F7	1	Allow semantic annotations for each process step.	Describe process element: annotate to a common semantic model
F12	1	Show smart process steps that can be included in the Process Designer. Each process step has to have defined input and output.	Discover partner services matching search criteria.
F13	1	Show the suppliers (service providers) that have completed the PLUG process and are ready for selection by a broker. (see also F5 and Mock-up L1-3).	Annotate process, bind services
F31	1	Changing suppliers should be possible during process execution, in case of any the delivery risk. This implies the possibility to re-optimize the process. (see also F41).	Annotate process activities and resolve (suggest) matching partner services bindings.
F47	2	It should be possible to manually change the partners associated with each activity and run new simulations;.	Describe process elements: change process activity bindings manually
F48	1	Define threshold value in order to have reference to the monitoring and the simulation components/functionalities on when to extract information.	Describe process elements: enter threshold values for monitoring
F49	1	Ability to search and reuse the existing templates processes and best practices (patterns) which are in the repository and then to take these and modify (See F50).	Browse and load process template (copies) to start from a predefined design.
F50	1	The system should allow tailoring the standard processes or design new manufacturing processes using a graphical interface and a standard notation.	Browse, edit and save (locate and manage) template processes
F51	1	It should be possible to design a new manufacturing process from scratch. .	Create new (empty/blank) process

F52	2	In the smart process design, the system must search and fetch potential partners on specific criteria (For example; Market sector; services type, Location, ...). Then it should be possible to drill down each partner in order to access more detailed information like Name, know-how, related services, technology used, ADVENTURE's trust index, etc.;.	Find and browse details for potential partners. (from Data Provisioning and Discovery)
F53	2	The system should enable the association of documents (specifications, drawings, manuals, etc.) To each of the activities of the collaborative process in order to share the vital information between the partners involved in each activity; This is not about content management or document flow, but simply being able to connect a document to a specific process step definition in the library.	Configure process elements: link/attach artifacts (e.g. Document) to activities.
F56	3	The system should permit the assignment of a set of preferential partners/suppliers for each outsourced activity (per partner). E.g. A group of preferred partners.	Configure process elements: bind partner service from favorites.
F62	2	Within the process design it must be possible to specify dependent tasks with parameters such that when executed (or not executed) - for example, so an alert can be sent when a process is executed - See also F63.	Configure process elements: define monitoring alert.
F69	4	Document Validation Process – e.g. Approval of documents.	Design approval process model
F79	1	Describe/specify restrictions/conditions/objective(s) (at design time) on non-functional properties for (whole) smart process so that designed processes can be further optimized.	Configure process model and process model elements: specify requirements, constraints, non-functional requirements for optimization (from Optimization)

Table 5 - Process Designer: Tasks to Functionalites Mapping

Max(P)	Tasks	Functionalities
1	Visualize the adoption	Display potential change Display alternative partners Display alert page
1	Visualize the partners search	Display the search results Display the search results in detail Display the status of suppliers
2	Visualize the partner management	Display a table to specify the required manufacturing capabilities Display the partner information
2	Visualize the partner's profile	Display a table to add data of partner
1	Visualize the process monitoring	Display the progress Display the stock level Display the process steps Display the status information of activities Display real-time information Display active processes Display alert page
1	Visualize the optimization	Display the optimized composition of services
1	Visualize the simulation	Display the simulation report
1	Visualize the forecasting	Display forecasting result
1	Visualize the process design	Display alternative/recommended partners
1	Visualize the process search	Display the search field Display comparison report
3	Visualize the documents management	Display the access right of documents
2	Visualize the message exchange between users	Display message to the user
	Visualize the configuration	Display configurable dashboard

Table 6 - Cloud Storage: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F1	1	During the PLUG process, partner information needs to be available through the dashboard. This can be static information, e.g., profile information, and dynamic information, e.g., current capacities.	Store profile information Provide profile information
F2	1	The Plug process is automatically started whenever a broker adds a new partner.	Provide profile information
F3	1	Allow Customers to add/enter data required by the PLUG process through the dashboard.	Store data Provide data

F5	2	Define the PLUG process, i.e., prerequisites of a supplier, in order to enable a broker to initiate a collaboration with a supplier.	Store data Provide data
F6	2	Provide information about min/max duration of the steps of the PLUG process.	Store data Provide data
F8	1	Abstraction, i.e., mapping and transformation, of partner interfaces.	Store Mapping Provide Mapping
F9	3	Allow Customers to express interest in a company products (for further contact). (relates to F10).	Store profile information Provide profile information
F10	1	Create a browsable catalog of ADVENTURE partners.	Storing the partner Providing the partners
F11	2	Enable a supplier to identify customers that initiated PLUG processes (even if not successful) with this supplier.	Maintain relationships history data
F12	1	Show smart process steps that can be included in the Process Designer. Each process step has to have defined input and output.	Maintain service descriptions
F13	1	Show the suppliers (service providers) that have completed the PLUG process and are ready for selection by a broker. (see also F5 and Mock-up L1-3).	Maintain service descriptions
F20	4	Define a policy if and how long to store data that is transmitted through ADVENTURE.	Maintain configured data retention and archiving policy and apply
F21	2	Clear visual representation of historic data.	Provide data
F22	2	Explore and display data granularity through dashboard.	Store data Provide data
F23	3	Presentation of interactive graphs / tables for data visualization are needed.	Provide data
F24	1	Rule's based message/alert/notification generation.	Store data Provide data
F26	2	Comparing forecast (done by the supplier) with prediction data (done by customer) to assess accuracy of the forecast.	Store data Provide data
F31	1	Changing suppliers should be possible during process execution, in case of any the delivery risk. This implies the possibility to re-optimize the process. (see also F41).	Maintain service descriptions

F34	2	Ability to save settings (virtual factories processes and partners) for future use.	Store processes Retrieve Processes
F35	4	Save content (e.g. Delivery dates) from information exchanged in process so that information can be extracted (e.g. Lead times).	Store data Provide data
F36	2	During the runtime phase all events must be recorded, including start and end of tasks time, alarms (e.g. Machine failed) and e.g. Partner suddenly not available or delayed two days. All these events come from the factories/partners.	Store runtime events
F43	1	After designing a smart process (e.g. During PLUG phase) then the following results should be stored: the configuration (the goal goals/metrics can be defined, partner information can be connected to, partner assignments, non-functional requirements (costs delivery times)). This should then enabled compare and find best configuration. Reports are generated via F44. Comparison is also a separate requirement.	Save processes and simulations
F44	2	At the end of a simulation a simulation report has to be displayed and saved. This should include details about the configuration parameters, goals etc.	Store data
F46	2	When a partner changes (See change F47 - automatic is not considered) it should be possible to do simulations taking into account the current data of process execution and data that are available for new potential partners. This is an extension of F43 since in this case the partner is being injected midway in to the process.	Store data
F47	2	It should be possible to manually change the partners associated with each activity and run new simulations;.	Store data
F49	1	Ability to search and reuse the existing templates processes and best practices (patterns) which are in the repository and then to take these and modify (See F50).	Maintain process templates
F50	1	The system should allow tailoring the standard processes or design new manufacturing processes using a graphical interface and a standard notation.	Maintain process templates

F52	2	In the smart process design, the system must search and fetch potential partners on specific criteria (For example; Market sector; services type, Location, ...). Then it should be possible to drill down each partner in order to access more detailed information like Name, know-how, related services, technology used, ADVENTURE's trust index, etc.;.	Maintain partner data
F53	2	The system should enable the association of documents (specifications, drawings, manuals, etc.) To each of the activities of the collaborative process in order to share the vital information between the partners involved in each activity; This is not about content management or document flow, but simply being able to connect a document to a specific process step definition in the library.	Store documents and their links to process steps
F54	2	The documents connected with process steps (or possibly the steps themselves) in the process step library should have access rights.	Authorized access to documents
F56	3	The system should permit the assignment of a set of preferential partners/suppliers for each outsourced activity (per partner). E.g. A group of preferred partners.	Maintain partner data
F57	2	The system should allow descriptions, photos, catalogs, attached to every partner. These documents are sent when a potential client asks for details about their activities, capabilities.	Maintain service descriptions
F59	2	All active processes should be viewable in the Dashboard.	Maintain process data
F60	3	The system shall provide information on past collaboration of partners with a party, so that that party can reutilizes these partners (e.g. Fetch a participation log from every ADVENTURE members).	Store partner and collaboration dataProvide partner and collaboration data
F62	2	Within the process design it must be possible to specify dependent tasks with parameters such that when executed (or not executed) - for example, so an alert can be sent when a process is executed - See also F63.	Maintain process dependency data

F63	2	Within the Process Execution Engine, if there are rules specified from a designed process which are broken, then an alert must be generated.	Provide process dependency data
F67	3	It should be possible to have version control of the documents associated to each smart process activity.	Provide versioning for documents
F68	4	Collaborate on a document through a process.	Store documents Provide documents
F70	1	Supplier side: Describe/specify offered manufacturing capabilities as services utilizing semantic annotation.	Maintain descriptions
F71	1	Customer side: Describe/specify required manufacturing capabilities as services utilizing semantic annotation.	Maintain descriptions
F72	1	Search/browse/query the data repository to find appropriate services (as offered by real factories) by entering a semantic description of services which needs to be exchanged.	Provide data based on descriptions
F73	1	Process monitoring data from smart objects is gathered.	Store monitoring data
F76	1	Process monitoring data from smart objects is stored in ADVENTURE's data repository.	Store monitoring data
F77	1	Show services appropriate for realizing certain process steps. Initiated manually to query the ADVENTURE system.	Maintain service descriptions
F78	1	Show non-functional properties (e.g. Default (typical) cost, delivery time, carbon footprint) of discovered services.	Maintain service descriptions
F80	1	Show/propose optimized composition of services for a specific smart process.	Maintain service descriptions
F83	3	Allow potential customers to identify former customers.	Maintain partner data

Table 7 - Cloud Storage: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Store profile information Store data Store Mapping Storing the partner Maintain relationships history data Maintain service descriptions Store processes Store runtime events Save processes and simulations Maintain process templates Maintain partner data Maintain process dependency data Store monitoring data Maintain process data Store partner and collaboration data	Store semi-structured data
1	Maintain descriptions	Store semantic data
1	Provide profile information Provide data Provide Mapping Providing the partners Provide processes Provide partner and collaboration data	Discover and provide specific semi-structured data
1	Provide data based on descriptions	Discover and provide specific semantic data
2	Store documents Store documents and their links to process steps	Store binary data
2	Provide documents	Discover and provide specific binary data
2	Store documents and their links to process steps	Links between different storage types
2	Authorized access to documents	Authentication system for access
3	Provide versioning for documents	Version control for binary storage
4	Maintain configured data retention and archiving policy and apply	Data archiving engine

Table 8 - Data Provisioning and Discovery: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F1	1	During the PLUG process, partner information needs to be available through the dashboard. This can be static information, e.g., profile information, and dynamic information, e.g., current capacities.	Search for static partner data Provide static partner data
F3	1	The Plug process is automatically started whenever a broker adds a new partner.	Search for static partner data – services descriptors Provide static partner data
F4	1	Allow Customers to add/enter data required by the PLUG process through the dashboard.	Provides data to ADVENTURE
F11	1	Allow semantic annotations for each process step.	Provide annotation models
F12	1	Abstraction, i.e., mapping and transformation, of partner interfaces.	Provide semantic models for annotations
F14a	1	Create a browsable catalog of ADVENTURE partners.	Create partner profile Store profile Browse/Search partner profile
F16	1	Show smart process steps that can be included in the Process Designer. Each process step has to have defined input and output.	Find a partner service activities (process steps) from the service descriptor
F17	1	Show the suppliers (service providers) that have completed the PLUG process and are ready for selection by a broker. (see also F8 and Mock-up L1-3).	Discover services
F38	1	Changing suppliers should be possible during process execution, in case of any the delivery risk. This implies the possibility to re-optimize the process. (see also F49)	Search for matching service provider
F56	1	Ability to search and reuse the existing templates processes and best practices (patterns) which are in the repository and then to take these and modify (See F57).	Annotate processes Store processes as templates Browse/Search templates

F57	1	The system should allow tailoring the standard processes or design new manufacturing processes using a graphical interface and a standard notation.	Maintain process templates
F76	1	Supplier side: Describe/specify offered manufacturing capabilities as services utilizing semantic annotation.	Describe own manufacturing services Provide description models to use Store own service descriptions Update own service descriptions Delete own service descriptions
F77	1	Customer side: Describe/specify required manufacturing capabilities as services utilizing semantic annotation.	Provide annotation models
F78	1	Search/browse/query the data repository to find appropriate services (as offered by real factories) by entering a semantic description of services which needs to be exchanged.	Browse services repository Query services repository Search services by template (example)
F84	1	Show services appropriate for realizing certain process steps. Initiated manually to query the ADVENTURE system	Query service operations matching process step model
F85	1	Show non-functional properties (e.g. Default (typical) cost, delivery time, carbon footprint) of discovered services.	Browse/query service NFP properties
F87	1	Show/propose optimized composition of services for a specific smart process.	Find process templates matching a (abstract) process step compliance rules Find service descriptions matching process step and process requirements and constraints
F8	2	Define the PLUG process, i.e., prerequisites of a supplier, in order to enable a broker initiating collaboration with this supplier.	Annotate process
F10	2	Provide information about min/max duration of the steps of the PLUG process.	Provide information about supplier
F15	2	Enable a supplier to identify customers that initiated PLUG processes (even if not successful) with this supplier.	Provide relationships history

F44	2	Ability to save settings (virtual factories processes and partners) for future use.	Save processes/partners annotations
F53	2	When a partner changes (See change F54 - automatic is not considered) it should be possible to do simulations taking into account the current data of process execution and data that are available for new potential partners. This is an extension of F50 since in this case the partner is being injected midway in to the process	Query for partner services and operations details required for simulation
F54	2	It should be possible to manually change the partners associated with each activity and run new simulations;	Find a partner service/operation
F59	2	In the smart process design, the system must search and fetch potential partners on specific criteria (For example; Market sector; services type, Location...). Then it should be possible to drill down each partner in order to access more detailed information like Name, know-how, related services, technology used, ADVENTURE's trust index, etc.;	Browse/Query partner descriptions
F65	2	The system should provide descriptions, photos, catalogs, attached to every partner. These documents are sent when a potential client asks for details about their activities, capabilities	Browse partner services descriptions and access accompanying documents
F71	2	Within the process design it must be possible to specify dependent tasks with parameters such that when executed (or not executed) - for example, so an alert can be sent when a process is executed - See also f71a.	provide process dependency data No
F71a	2	Within the Process Execution Engine, if there are rules specified from a designed process which are broken, then an alert must be generated	Provide process dependency data No

Table 9 - Data Provisioning and Discovery: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Search for static partner data Search for static partner data – services descriptors	Find ADVENTURE partners Find static data Find dynamic data Find ADVENTURE services Find services for activity Search by QoS criteria Browse service details
1	Provide static partner data.	Create profile Edit profile Save profile
1	Provides data to ADVENTURE	Create profile Describe service Describe software service Describe smart object service Describe human service
1	Provide annotation models	Classify service products Annotate
1	Provide semantic models for annotations	Annotate
1	Create partner profile Store profile	Create profile Edit profile Save profile
1	Browse/Search partner profile	Find ADVENTURE partners Browse my partners Browse profile details
1	Find a partner service activities (process steps) from the service descriptor	Find services for activity Browse service details
1	Discover services	Find ADVENTURE services Find services for activity
1	Search for matching service provider	Find services for activity
1	Annotate processes	Annotate
	Store processes as templates	Save as process template

	Browse/Search templates	Find process templates Browse process details
1	Maintain process templates	Save as process template Find process templates Edit process model
1	Describe own manufacturing services Provide description models to use Store own service descriptions Update own service descriptions Delete own service descriptions	Describe service Describe software service Describe smart object service Describe human service Manage attachments Add service Remove service Annotate
1	Browse services repository Query services repository Search services by template (example)	Find ADVENTURE services Find services for activity Search by QoS criteria
1	Query service operations matching process step model	Find services for activity
1	Browse/query service NFP properties	Search by QoS criteria
1	Find process templates matching a (abstract) process step compliance rules	Find process templates Browse process details
1	Find service descriptions matching process step and process requirements and constraints	Find services for activity Search by QoS criteria
2	Provide information about supplier	Create profile Edit profile Save profile Describe service
2	Provide relationships history	Browse my partners Find process
2	Save processes/partners annotations	Annotate
2	Query for partner services and operations details required for simulation	Find ADVENTURE services Find services for activity Search by QoS criteria Browse service details
2	Find a partner service/operation	Find ADVENTURE services Find services for activity Search by QoS criteria

		Browse service details
2	Browse/Query partner descriptions	Find ADVENTURE partners Browse my partners Browse profile details
2	Browse partner services descriptions and access accompanying documents	Browse service details

Table 10 - Message Routing: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F15	1	Securely validate and transmit data to customer.	Secure communication / messaging
F16	1	Buffer data in case partner is not available.	Delayed messaging
F27	1	Connectors from legacy software to ADVENTURE.	Provide and register Gateways
F28	1	Enable the broker to compute ATP ("available to promise") or CTP ("capable to promise") based on available resources, i.e., availability of material and capacity within the factory and also the availability of materials from suppliers.	Deliver ATP via Gateways
F29	1	Enable the broker to gather dynamic, real-time information about, e.g., material stock position, capacity position, etc. (see also F28: this information will support the computation of ATP/CTP).	Deliver ATP via Gateways
F30	1	The broker should be able to see on the Dashboard the stock levels at own factory, in transit, and at suppliers for any material.	Deliver inventory data via Gateways
F37	1	The system must allow monitoring of process steps, providing status information of the activities in each of the partners (not started, late in x hours, in production, in testing, in packaging, in shipping, finished, % conducted) via the Dashboard. This information would come from F36 but maybe a subset of information (e.g. not all events - only important ones - e.g. started,	Routes event information stored in cloud to Process Execution Engine

		errors etc.).	
F61	1	ADVENTURE should be able to integrate with legacy systems such as MES and ERP.	Gateways will do it as appropriate
F33	2	The system must issue alerts (alert page) in case stock levels cross agreed thresholds (relates to F41).	Route the alert
F19	3	Manage several data formats and transformations between them.	Takes care of data transformations if necessary
F32	3	Notify partners about intended orders and processes with estimated definitions of product type, quantity and delivery date.	Deliver upcoming order
F85	3	Implement instant messaging for partners collaborating in a virtual factory integrated into the ADVENTURE system.	Instant Messaging functionality

Table 11 - Message Routing: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Secure communication / messaging	Provide secure communication
1	Delayed messaging	Provide message buffering
1	Provide and register Gateways Deliver upcoming order	Provide catalog of registered components by role Enable component registration Provide integration for components (e.g. Gateways,…)
1	Deliver upcoming order Deliver ATP via Gateways Deliver inventory data via Gateways - Routes event information stored in cloud to Process Execution Engine	Deliver messages between components
1	Route the alert	Deliver messages from a component (e.g. Process Monitoring) to users
3	Takes care of data transformations if necessary	Transformation of data will be handled by the Transformation Services component

3	Instant Messaging functionality	Provide Instant Messaging functionality for users

Table 12 - Transformation Services: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F8	1	Abstraction, i.e., mapping and transformation, of partner interfaces.	Data transformation service
F18	3	Update data formats if necessary.	Take care of format/ message transformations
F19	3	Manage several data formats and transformations between them.	Take care of data transformations

Table 13 - Transformation Services: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Data transformation service	Provide data transformation
3	Take care of format/ message transformations	Provide format mapping
3	Take care of data transformations	Provide data transformation

Table 14 - Gateways: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F27	1	Connectors from legacy software to ADVENTURE.	Provide and register gateways
F28(*)	1	Enable the broker to compute ATP ("available to promise") or CTP ("capable to promise") based on available resources, i.e., availability of material and capacity within the factory and also the availability of materials from suppliers.	Deliver ATP via gateways
F29(*)	1	Enable the broker to gather dynamic, real-time information about, e.g., material stock position, capacity position, etc. (see also F28: this information will support the computation of ATP/CTP).	Deliver ATP via gateways
F30(*)	1	The broker should be able to see on the Dashboard the stock levels at own factory, in transit, and at suppliers for any material.	Deliver inventory data via gateways

F61	1	ADVENTURE should be able to integrate with legacy systems such as MES and ERP.	Integrate with legacy systems such as MES and ERP
F32 (*)	3	Notify partners about intended orders and processes with estimated definitions of product type, quantity and delivery date.	Deliver upcoming order

(*) These requirements are related to gateways because of the fact that this kind of information flows through them, but gateways are not responsible of gathering/providing this specific data, just interfacing the services that provide them. So F32, F30, F29 and F28 can be seen as particular scenarios of F61.

Table 15 - Gateways: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Provide and register gateways	Provide gateway for specific external system Provide registration and description functionality for gateway.
1	Integrate with legacy systems such as MES and ERP	Provide gateway for specific external system Send messages to ADVENTURE network Invoke transformation service

Table 16 - Process Execution: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F2	1	The Plug process is automatically started whenever a broker adds a new partner.	Load plug process for partner Instantiate plug process Execute plug process

Table 17 - Process Execution: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Load plug process for partner	Handle process execution
1	Instantiate plug process	Handle process execution
1	Execute plug process	Handle process execution

Table 18 - Process Adaption: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F31	1	Changing suppliers should be possible during process execution, in case of any the delivery risk. This implies the possibility to re-optimize the process.(see also F41).	Adapt process model Adapt process instance model

Table 19 - Process Adaption: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Adapt process model	Enable changing of suppliers during process execution
1	Adapt process instance model	Enable changing of suppliers during process execution

Table 20 - Forecasting and Simulation: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F25	1	Supplier forecast data giving information on processes such as stock control.	Fetch forecast data from supplier Present forecast data from supplier
F26	2	Comparing forecast (done by the supplier) with prediction data (done by customer) to assess accuracy of the forecast.	Fetch forecast data from supplier Fetch forecast data from customer
F28	1	Enable the broker to compute ATP ("available to promise") or CTP ("capable to promise") based on available resources, i.e., availability of material and capacity within the factory and also the availability of materials from suppliers.	Fetch necessary data
F39	3	Collect forecast from customer side (i.e. more like recognizing trends). Note: This is not the same as process related forecasting and hence lower priority. Main requirement from ABB but e.g. AZEV could find it useful.	-Fetch forecast data from customer

F40	3	View the sales trends and equipment investment trends for a set of selected companies. (see also F39).	Fetch necessary data
F44	2	At the end of a simulation a simulation report has to be displayed and saved. This should include details about the configuration parameters, goals etc.	Generate report Return report
F45	2	After a process simulation, the ADVENTURE simulation component should be able to do "what-if" analysis supported by specific changes on the smart process.	Read simulation details from Cloud Storage Generate report Return report
F46	2	When a partner changes (See change F47 - automatic is not considered) it should be possible to do simulations taking into account the current data of process execution and data that are available for new potential partners. This is an extension of F43 since in this case the partner is being injected midway in to the process.	Trigger start of simulation Save simulation details to cloud storage Generate report Return report
F81	3	A means to assess the environmental impact of contracts with potential partners. I.e. Process steps that are in a category environment, and allow fetching of information about hazardous material (Mockup L1-1).	Trigger start of simulation Save simulation details to cloud storage Generate report Return report
F82	3	A means to compare potential partners regarding the environmental impact of a contract. I.e. fetch information for various partners for comparison in a table.	Trigger start of simulation Save simulation details to cloud storage Generate report Return report

Table 21 - Forecasting and Simulation: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Fetch forecast data from supplier	Handle supplier forecasts Compare forecasts

1	Present forecast data from supplier	Handle supplier forecasts
2	Fetch forecast data from customer	Enable forecast comparison Collect forecast data
1	Fetch necessary data	Enable computation of ATP/CTP Enable computation of sales trends
2	Generate report	Provide simulation report
2	Return report	Provide simulation report
2	Trigger start of simulation	Provide process simulations Provide assessment of environmental impacts
2	Save simulation details to cloud storage	Provide process simulations Provide assessment of environmental impacts

Table 22 - Monitoring: Requirements to Tasks Mapping

Req	P	Requirement	Tasks
F24	1	Rule's based message/alert/notification generation.	Rules apply monitoring data
F30	1	The broker should be able to see on the Dashboard the stock levels at own factory, in transit, and at suppliers for any material.	Monitor stock level
F33	2	The system must issue alerts (alert page) in case stock levels cross agreed thresholds (relates to F41).	Real-time monitoring
F36	2	During the runtime phase all events must be recorded, including start and end of tasks time, alarms (e.g. machine failed) and e.g. partner suddenly not available or delayed two days. All these events come from the factories/partners.	Logs all process events

F37	1	The system must allow monitoring of process steps, providing status information of the activities in each of the partners (not started, late in x hours, in production, in testing, in packaging, in shipping, finished, % conducted) via the Dashboard. This information would come from F36 but maybe a subset of information (e.g. not all events - only important ones - e.g. started, errors etc.).	Monitor process activities status
F38	1	The system shall present the status of all process steps in a graphic manner. (e.g., by color in each task in the process, including status information and notes related to it). This builds of F37 to provide further UI improvements.	Provide UI for process statuses
F41	1	The system must issue alerts (alert page) in case of unexpected situations that can arise during the production process: delays in tasks and their reasons (problem on a partner machine, shipping delays, goods damage, etc.).	Continuous monitoring and active notification
F42	3	Extending alerts (F41) to include email/SMS/Mobile.	Continuous monitoring and active notification
F46	2	When a partner changes (See change F47 - automatic is not considered) it should be possible to do simulations taking into account the current data of process execution and data that are available for new potential partners. This is an extension of F43 since in this case the partner is being injected midway in to the process.	Provide process monitored data (i.e. already stored)
F48	1	Define threshold value in order to have reference to the monitoring and the simulation components/functionalities on when to extract information.	Process Monitor to reference parameters
F59	2	All active processes should be viewable in the Dashboard.	Monitor process state
F66	2	Process monitoring should support SMART objects with single or multiple sensors.	Process Monitoring
F73	1	Process monitoring data from smart objects is gathered.	Process monitoring data for consumption

Table 23 - Monitoring: Tasks to Functionalities Mapping

Max (P)	Tasks	Functionalities
1	Rules apply monitoring data Process Monitor to reference parameters active notification	Provide means to define rules to be evaluated; Trigger alerts, notifications and messages based on rules evaluation.
1	Monitor process activities status Monitor stock level Real-time monitoring Continuous monitoring Monitor process state Process Monitoring	Real-time data monitoring; Visualize process state Visualize task state and its details
2	Logs all process events	Log monitored data;
1	Provide UI for process statuses	Provide UI for data monitoring; Provide UI to configure data monitoring rules.
2	Provide process monitored data (i.e. already stored)	Provide Process Analytics data. Provide Process Historical data.
1	Process monitoring data for consumption	Define expressions to calculate key performance indicators based on spread data.

Table 24 - Smart Objects Integration: Requirements to Tasks Mapping

ID	P	Functional Requirement	Tasks
F66	2	Process monitoring should support Smart Objects with single or multiple sensors.	SMART Object integration for integration of multiple sensors Process Monitoring
F73	1	Process monitoring data from Smart Objects is gathered.	Maintain monitoring data Provide monitoring and aggregation of data Process monitoring data for consumption
F74	4	Process monitoring data from Smart Objects is locally pre-processed.	Pre-process monitoring data (aggregation, filtering) locally when possible
F75	1	Communication with Smart Objects, which monitor data which may be used in a process is enabled. They are connected through ADVENTURE enabling gateways as per other legacy	Provide interface to ADVENTURE to integrate smart objects as IT providers

		systems.	
F76	1	Process monitoring data from Smart Objects is stored in ADVENTURE's data repository.	Maintain monitoring data from smart objects Store monitoring data

Table 25 - Smart Object Integration: Tasks to Functionalities Mapping

Max (P)	Tasks	Functionalities
2	SMART Object integration for integration of multiple sensors	Message transformation Transmit message
2	Process monitoring	Transmit message
1	Maintaining monitoring data	Transmit message
1	Provide monitoring and aggregation of data	Message transformation Transmit message
1	Process monitoring data for consumption	Message transformation

Table 26 - Optimization component: Requirements to Tasks Mapping

ID	P	Requirement	Tasks
F79	1	Describe/specify restrictions/conditions/objective(s) (at design time) on non-functional properties for (whole) Smart Processes so that designed processes can be further optimized.	Specify constraints on non-functional properties Specify objective(s) on non-functional properties
F80	1	Having modelled a Smart Process and specified restrictions and objectives, the ADVENTURE Broker wants to know, which partner services among alternative ones he should select.	Specify process model Identify candidate partner services Optimize process model

Table 27 - Optimization component: Tasks to Functionalities Mapping

Max(P)	Tasks	Functionalities
1	Specify constraints on non-functional properties Specify objective(s) on non-functional properties Specify process model Identify candidate partner services	Provide optimization problem
1	Optimize process model	Solve optimization problem