D3.1 Global Architecture Definition Document

Authors: ASC (lead), TIE, UVA, UVI, ISOFT

Delivery Date:	Due Date:	Dissemination Level:
2012-04-04	2012-03-31	Public

This document is the first technical deliverable of the ADVENTURE project. The global architecture deliverable is used to define components and their interaction. It is based on the main components of the description of work but it is performed at a higher level of detail and also involves the specification of sub-components and actors.

Document History	
Draft Version	V0.1, ASC, October 20th, 2011 V0.4, ASC, February 6th, 2012 V0.5, ASC, March 22nd, 2012 V0.6, ASC, March 23rd, 2012 V0.7, ASC, March 28th, 2012 V0.8, ASC, April 2nd, 2012 V0.9, ASC, April 3rd, 2012 V1.0, ASC, April 4th, 2012
Contributions	ISOFT - Georgi Pavlov TUDA - Dieter Schuller UVI - Jürgen Mangeler TIE - Lennert Kujipers ASC - Rafael Karbowski - Sven Abels - Tim Dellas
Internal Review 1	Dieter Schuller, TUDA
Internal Review 2	Gash Bhullar, TANet
Submission of Final Version	April 4th, 2012

Table of Contents

Executive Summary

This document describes the architecture of the ADVENTURE Framework and gives an introduction to the tasks and the functionality of the components and how they belong together. It will be used to give an overview of the framework and serves as basis for D3.2 Functional Specification and D3.3 Technical Specification.

First, the results of D2.1 (Vision Consensus Document) and D2.3 (Requirements Document) are shown. Afterwards, an overview of the platform is given, also detailing the impact that the planned architecture has on the system.

The platform contains 13 components that are arranged in four layers, which are described in section 4. Those four layers are

1. Data Management,
2. Process,
3. Data Exchange and
4. Interaction layer.

The Data Exchange layer contains the Message Routing component that will be used by all other components to communicate. All data will be stored in the cloud using the Cloud Storage component, which belongs to the Data Management layer. The Interaction layer contains all components that will be used to interact with users or external systems. All components that are related to processes are located in the Process layer, except the Process Designer, which is a user interface.

Finally, section 5 details which risks might be occurring in the further pursuit of the project and details what should be addressed to prevent problems.

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. Within this deliverable components and their interaction are defined.

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 goal of this deliverable is to shape a technical understanding of the vision and the associated requirements. As such, the architecture definition kicks off the technical work of the project and is therefore the first deliverable of WP3. It defines

and describes the global architecture that is employed in the project and that is used by the functional and technical specifications as a base for their component descriptions.

1.3 Document Status

This document is listed in the DOW as public since the architecture may be used by external parties as a reference model or as a base for understanding the technical elements of the project. It may also be used as a starting point when integrating external systems to ADVENTURE, e.g. ERP systems or value added networks.

1.4 Target Audience

This deliverable is to be used by all participating project members. It will define a high-level architecture as well as a set of core components for the project. As such, this deliverable will form the baseline for the functional and technical specification.

1.5 Abbreviations and General Terms

A definition of general, 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

1.6 Document Structure

This deliverable is broken down into the following sections:

Section 1 (this section) presents the context of the document.

Section 2 describes the vision of the project consortium from an architectural point of view and describes which requirements have to be taken into account for creating the architecture of the system.

Section 3 gives a comprehensive view on the system and its architecture. It describes the whole system with its components. It elaborates how the components interact with each other and it describes the features which some of the components rely on.

Section 4 describes the ADVENTURE components in a more detailed way. This includes the definition and description of the components as well as the integration into the platform.

Section 5 shows potential risks in terms of architectural choices and how to minimize them. This section will also make predictions on possible problems that the consortium has identified during the.

Section 6 finally gives an introduction into the next steps and evaluates the status of the architecture.

2 Impact of Vision and Requirements

To set this document into context, this section revisits the Vision Consensus Document (D2.1, see section 2.1) and the Requirements Analysis Document (D2.3, see section 2.2). It presents, which changes from the initial Description of Work (DOW) have to be implemented in the Global Architecture Definition.

2.1 Vision

The goal of ADVENTURE is to simplify the establishment, management, adaptation and monitoring of dynamic manufacturing processes in Virtual Factories. This includes the finding of partners, the design, forecasting and simulation of Smart Processes, and their execution and real-time monitoring. Within the Vision Consensus Document, the consortium voiced visions about ADVENTURE and use cases and scenarios that ADVENTURE is going to realize. This section examines D2.1 from a architectural perspective to emphasize what has to be defined in the global architecture specification to reach these goals.

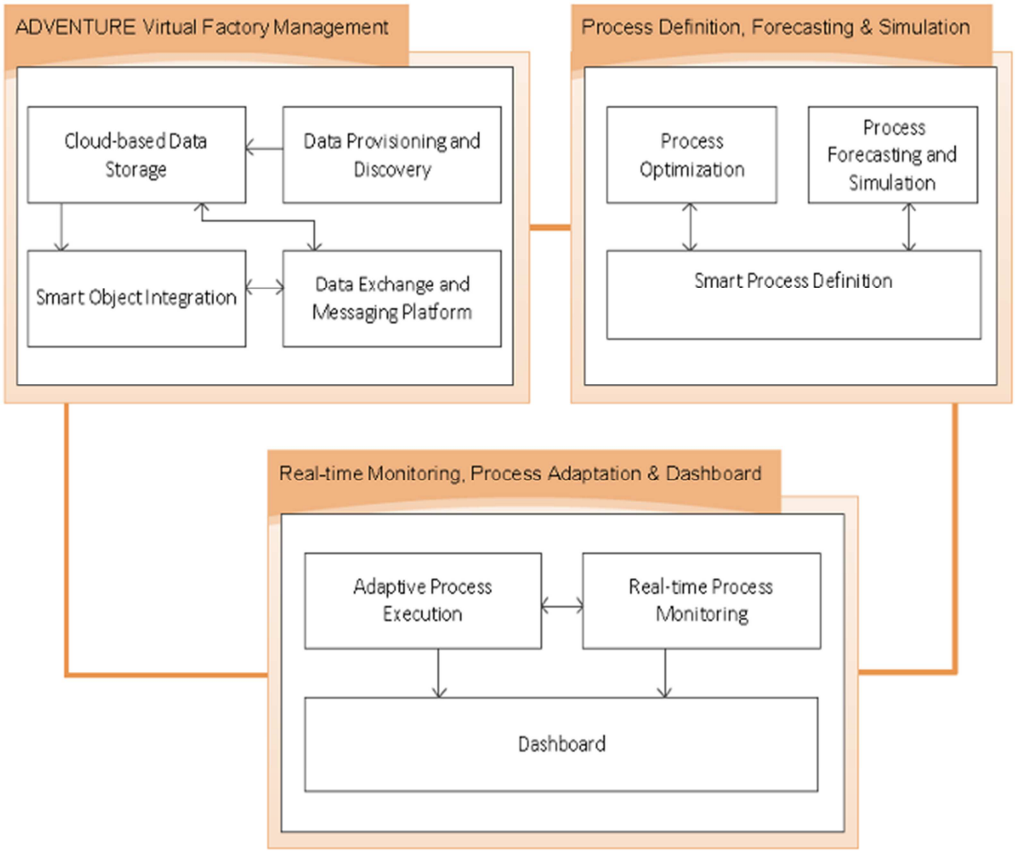

Figure 1: ADVENTURE components

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 his company, products, services and processes. To achieve this in a user-friendly way, ADVENTURE has to provide an editor in the

scope of the **Data Provisioning & Discovery** component to enter, view, update or delete this data. For reasons of availability, accessibility, access-control and the possibility to have redundant backups if needed, this data will be stored in the cloud. The **Cloud Storage** component should support several types of data storage like semi-structured data storage (for example for XML or JSON data) used internally by the ADVENTURE Platform, as well as semantic data necessary for semantic company descriptions and also data storage for binary files. Binary files may be used for storing documents such as specifications or even multimedia files.

To define Smart Processes ADVENTURE has to provide a **Process Designer**. To improve and facilitate the usability of ADVENTURE, in D2.1 all user interfaces were described to be accessible via a single application interface with a single look and feel and a quick learning curve. All the user interfaces will therefore be embedded in the **Dashboard**, including the process designer.

Within the **Optimization** component an optimization model for manufacturing processes including constraints modeling will be conceptualized and formalized. This will be performed in order to establish a basis for the optimization step. Depending on specific optimization needs for certain processes, appropriate optimization techniques will be implemented and applied.

Forecasting & Simulation of the process is the execution of the process in forecast mode, watching for the forecast / simulated results and showing them to the user. The Forecasting & Simulation is closely related with the **Process Execution** component, which usually executes process models. Thus, the forecasting is also based on the Process Monitoring to visualize results. Optimization, Simulation, Monitoring, Forecasting and Process Execution are interrelated, but have been split into separate components to keep the platform modular and flexible – which will also help the consortium split the RTD tasks in a more efficient way.

The **Real-time Process Monitoring** which is initially described in the vision document shows the actual status of the process execution and can additionally query Smart Objects like sensors etc. for their current state.

As already hinted at, a lot of messaging between different components and, in some cases, even between different instances of one component is necessary, when a component can be instantiated multiple times in the cloud. To ensure a simple and standardized communication, a **Message Routing** component is needed which connects the different components and which cares for the connection of components. Thus, these components don't need information about URLs or locations of other components to perform their work. The vision consensus document also includes scenarios and thoughts about the integration of external systems, legacy systems and smart objects.

Smart Objects will be integrated via the **Smart Object Integration** component, which will communicate with the platform via the Message Routing. External systems like legacy systems etc. should also be able to communicate with the ADVENTURE Platform's Message Routing component making use of the **Gateway** component, which effectively fulfils the role of a bridge, connecting the external system with the ADVENTURE platform. The Gateway as well as the Message Routing may invoke **Transformation Services** that can be used to translate between external (legacy) technology which will use different messaging protocols, interfaces and message

formats. The Gateways therefore will be the only components that might need to be expanded or recreated when a new ADVENTURE Member wants to connect uncovered legacy systems to the Message Routing. The Message Transformation may be used as a base by the Gateways to transform a variety of data formats, hence allowing ADVENTURE a wide support of systems.

2.2 Requirements

Deliverable D2.3 defines the requirements of the project. These requirements can be seen as the basis for the functional specification D3.2 and also present a direct input to D3.3, but they bear few implications for the Global Architecture Definition, and those are in an expected scope. The list of requirements for an ADVENTURE implementation is presented in D2.3 both in the form of UI-related mock-ups and their underlying functions, which has some implications on the features offered by the user interfaces of components that directly present a UI or indirectly provide functionality or data in the UI.

The complete requirements list in D2.3 list has initially been collected from the use case companies and then been completed by the other project partners. Those requirements are related to the original component names (cf. Section 3.2.2 for more information) and their impact on D3.1 can be summarized as follows:

1. *Data exchange and messaging platform between partners*: The requirements define the need to ensure the smooth running of operational activities (e.g. order handling, inventory management, necessary notification, etc.) between partners. A common messaging platform should be developed for mapping the legacy systems and to guarantee secure communication between virtual factory partners in a holistic solution. This communication platform ensures the information (data and message) monitoring in terms of both macro level (e.g. ordering process) and micro level (e.g. query of all suppliers buffer levels). The definite desired support for data reporting of many factory level data details forced the separation of the messaging platform into Message Routing, Gateways and Transformation Services.

2. *Real-time data provisioning and process monitoring*: The list of requirements indicates demands for both synchronous and asynchronous data to be able to serve as a basis for data discovery in the Data Provisioning & Discovery component. Many requirements also target the manual control Process Execution, Adaption and the UI presented for real-time monitoring. The rest of the requirements in this area detail the data transfer between partners. This is needed in order to be able to monitor a Smart Process.

3. *Cloud-based data storage*: According to the requirements, it is crucial to collect and store all data which is needed across component and factory borders in a cloud-based data storage system, including data collected from smart objects. Internally the cloud based data storage may be based on different storage

backend systems, e.g. for storing semantic, structured or semi-structured information. However, this will be hidden from the other components by providing a unified storage interface, so that other components don't have to directly manage the interaction with backend systems. The cloud data storage should store the necessary provisioning data, and all data needed for process forecasting and simulation purposes, which effectively means it should be the storage container for Smart Process models.

4. *Process visualization (Dashboard)*: It is required to provide a GUI - called "Dashboard" - to the virtual factory partners. This Dashboard primarily includes views for the Smart Process Designer and the Monitoring that makes the process status of the Adaptive Process Execution Engine visible. It also supports the functionalities that are necessary for the operational activities within the virtual factory. Additionally, it enables the configuration of notifications or alerts of possible deviations within a process.

5. *Modular User Interface*: The Dashboard should be configurable based on role of the viewing user. It should provide the necessary interaction (through plug-ins) and integration with the legacy systems between the virtual factory users through standard visualization mechanisms such as activity streams, messages, simple graphs, etc. Moreover, it should provide a mechanism (e.g. plugin-based) to support specialized partner-specific visualisation UI, if a company needs those.

6. *Modular architecture*: The components of the architecture should have well-defined interfaces, message formats, message protocols and discrete functions so that the components can theoretically be replaced by other solutions easily. This is important especially if the commercial solutions of consortium partners (especially TIE and C2K) are adapted or employed directly as it must be possible to use ADVENTURE without commercial products.

3 ADVENTURE Framework Overview

3.1 Abstract

As shown in Figure 2 the ADVENTURE platform is based on a distributed architecture, which consists of several components in 4 architectural layers, connected by a Data Exchange layer.

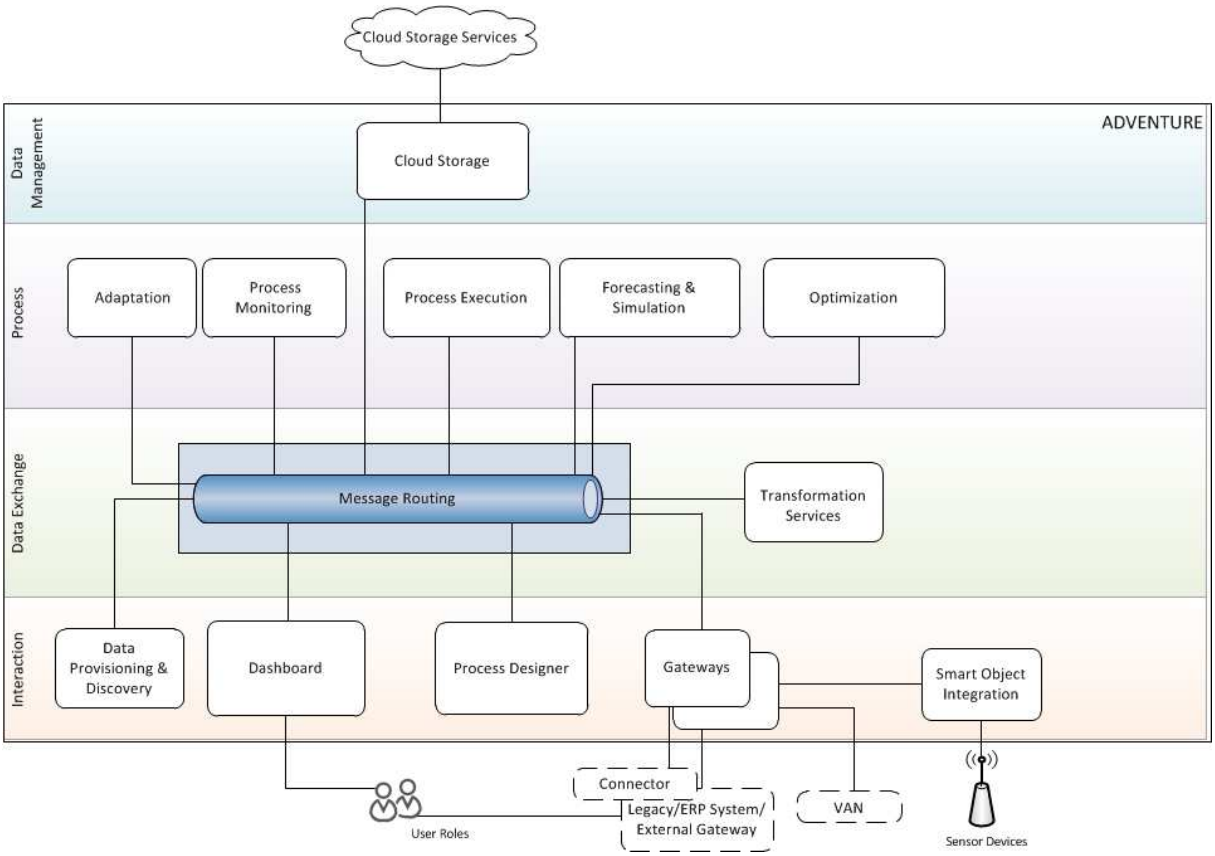

Figure 2: Architecture of the ADVENTURE platform

The architecture is categorized into 4 layers:

1. *Data Management*: This layer provides a persistence layer for ADVENTURE global resources. It contains the Cloud Data Storage service component and its modules that realize the storage of different data types.

2. *Process*: This layer hosts all the components that interact with Smart Processes.

3. *Data Exchange*: This layer provides platform wide communication services.

4. *Interaction*: This layer contains the components that interact with users and external systems, like legacy systems, VANs (value added networks), ERP systems and smart objects.

The components can be deployed on the same as well as on several physically separated systems, which includes the possibility to run several instances of one component on several systems simultaneously. It is possible to deploy the systems on webservers, virtualized in a cloud system or both. Which modes exactly are supported is to be defined in the Technical Specification (D3.3). The detailed description of the components can be found in Section 4. In the following, a short overview about the tasks of the components is provided.

The diverse components are connected by the Message Routing component. The communication with Smart Objects, legacy systems, ERP systems and other VANs will be done with dedicated and lightweight Gateway-components. These can apply existing Transformation Services to do the message transformations, thus the transformation implementation can be shared for different Gateways.

The user will interfere with the platform mainly with three components: The Data Provisioning component, the Process Designer and the Dashboard, which also hosts the other two components.

The Cloud Storage component will provide an abstraction for cloud-based data storage of binary files, relational data, structured information, and highly structured (semantic) data.

The actual execution of the designed Smart Processes will be managed by the Process Execution component. The user will be informed by the event-driven Monitoring component, and the Forecasting component will basically analyse the runtime data and provide decision support to the user. The Optimization component can analyse given processes and optimize them.

3.2 Changes from the DoW

Figure 1 shows the original components envisioned in the DoW. As already depicted in section 3.1, some refinements were applied to the architecture, which are summarized in this section.

3.2.1 Split of components

Some components have been split into multiple components to better enable working on the more narrow defined functionality and to have a more loosely coupled overall platform. The “Data Exchange and Messaging Platform” component has been split into three components:

- Message Routing
- Transformation Service
- Gateways

The “Adaptive Process Execution Engine“ component has been split into

- Execution
- Adaption

3.2.2 Renaming

In order to simplify naming, some components have been renamed. The new names are shown in Table 1.

Original component name	New component name
Cloud based Data Storage	Cloud Storage
Smart Process Definition	Process Designer
Adaptive Process Execution	Process Execution
Adaptive Process Execution	Adaption
Real-time Process Monitoring	Process Monitoring
Process Optimization	Optimization
Process Forecasting and Simulation	Forecasting & Simulation
Data Provisioning and Discovery	Data Provisioning & Discovery
Smart Object Integration	Smart Object Integration
Data Exchange and Messaging Platform	Message Routing
Data Exchange and Messaging Platform	Transformation Services
Data Exchange and Messaging Platform	Gateways

Table 1: Renaming and splitting of the original components

3.3 Cloud Architecture

The type of distributed architecture applied for ADVENTURE offers many advantages compared to a centralized approach.

- Due to easily expandable computing power this architecture has a high degree of scalability. Components that need a high amount of processing power in short time can be virtually duplicated easily by the project for achieving load balancing.
- The use of cloud-based data storage also implies a high scalability in terms of data throughput and storage capacity. This allows the ADVENTURE team to rely on a number of cloud storage providers to ensure a highly elastic data storage.
- Duplicated deployment on several systems can be used to guarantee a higher availability of the components. If one component unexpectedly fails, another instance can still do the task at hand (Failover).
- High protection against data loss, because the data can be stored redundantly on distributed systems if needed (Failover).

The communication of the components will be routed via the Message Routing component, so the different components don't need data about the location of their respective messaging partners, but only need set up with the messaging component that cares for the correct handling of the communication in the platform. Additionally, the Message Routing component can provide secure messaging, constant connections and push-notifications, which means that no polling techniques are needed.

To communicate with external (i.e. legacy) 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. One Gateway has to be created for each legacy system type, in order to translate the message protocol and message structure of the legacy system to those used internally in ADVENTURE. For the diverse systems in use, this is the only way for communication from ADVENTURE with the different legacy systems and vice versa, which can be accomplished. Technical possibilities to provide legacy system-to-connector interaction include TIE Smart Bridge and Control 2K IndrustreWeb that are being provided by the project partners TIE and Control 2K. Those solutions as well as open source solutions will be made available in the form of the Transformation Services component that can be used by the Gateway components to reuse existing technology.

3.3.1 Failovers

Cloud providers normally offer great failover opportunities e.g. by redundant replicas, backup services, image pools, etc. That on one hand allows delegation of part of or all of these critical operations and on the other hand, to increase significantly the reliability of the solutions for an affordable price.

To take advantage of that characteristic, it is important that the system and component design is harmonized with the failover features and specifics of the cloud provider that serves it. Depending on the cloud type (IaaS, Paas, SaaS) this may require some significant work (IaaS) or be more or less transparent (PaaS,SaaS). It is up to the system designer to come up with a strategy how to utilize the out-of-the box cloud features in this respect, and ultimately to design for architecture that survives disasters, taking the most of the underlying platform provider.

As a rule of a thumb, good cloud architectures should be impervious to reboots and re-launches. Some other concerns to consider in the cloud system design for failovers are:

- What happens if a node fails?
- How do you recognize a failure has occurred?
- How to replace a node?

- Which are the single point of failures and what happens when they fail (e.g. a load balancer)?
- How do you switch to the slave and synchronize state in master/slave scenarios?

A generic strategy at addressing such concerns could be to:

1) Have a coherent and automated backup and restore strategy for data
2) Build process threads that resume on reboot
3) Allow the state of the system to re-sync by reloading messages from queues (delayed messaging)
4) Keep pre-configured and pre-optimized virtual images to support (2) and (3) on launch/boot
5) Avoid in-memory sessions or stateful user context, move that to data stores.

3.3.2 Decoupling

The cloud paradigm reinforces the SOA principle for loose coupling. It is critical to build system design without tight dependencies between components to take full advantage of the inherent scalability of the cloud platform. The goal is that a "misbehaving" component (dead/slow in response/blocked) does not affect fatally the system. A key principle in the decoupling of system components to a great extent is to utilize asynchronous communication pattern between them.

Some things to be considered are:

- Which are the system components that can run as standalone processing units?
- How to introduce multiple instances of these components to scale horizontally and without compromising the good operational state of the component?
- How to introduce an asynchronous interaction with the component?

A common approach at realizing a loosely coupled system is by means of *messaging queues*. A message queue is a commodity service by now in most cloud providers. If a queue/buffer is used to connect any two components together, it can support concurrency, high availability and load spikes. As a result, the overall system continues to perform even if some components are momentarily unavailable. If one component dies or becomes temporarily unavailable, the system will buffer the messages and get them processed when the component comes back up (hence the design consideration for failovers).

3.3.3 Elasticity

Generally speaking, elasticity in the cloud comes in two flavors: proactive or triggered by environment.

Proactive elasticity is the capability of the system to scale based on expected load or schedule. It can be further split conceptually into to cyclic (regular), i.e. daily, nightly, weekly, monthly, etc and event driven, i.e. there's a scheduled event that will expectedly trigger load/unload of the system (holidays, release test period, etc.).

Elasticity triggered by the environment (or auto-scaling in Amazon terms) is a feature that scales your systems automatically and on unexpected demand, like load spikes (e.g. a new commercial for your product was released and everyone hits a request at your system to see what it's all about). Normally, cloud providers monitor the state of the systems and their environment and react automatically on dynamic changes in certain patterns and triggers.

Elasticity is very tightly related to the cost optimization and scalability. This feature actually ensures that you don't pay for more computing resources than you need and that you have sufficient computing resources exactly when you need them. The most essential consideration to realize this is that system nodes are automatically powered up and down in an operations efficient manner and transparently plugged in and out of the system. With that respect some concrete measures to ensure elasticity in the system design are:

- Automate your infrastructure: script the installation and configuration process
- Bootstrap your instances: synchronize state, step in the required role and plug in the interaction pattern with the other components

3.3.4 Parallelism

One of the benefits in using a cloud is the ease of introducing elasticity via parallelism. You can construct a cluster of nodes to meet high computation demands quickly and then reduce it as appropriate just as easy. A key prerequisite to do that smoothly is the automation of the elasticity and on the other hand – the design for parallelism in the system components internally. Generally, a component design considering thread safety and a share-nothing policy should serve the purpose to provide parallelism-safe design well.

On a system level, a topology with a load balancer redirecting requests to asynchronous processing nodes (e.g. in a web application scenario or master/slaves scenario) is a known best practice.

3.3.5 Reduction and Optimization of Network Communication

It's a generally good practice to reduce network communication and considering the cloud payment models that charge based on net traffic, I/O and CPU load it is even more important to design for optimal usage of resources for which you are charged.

Things to consider in Task 3.3 are:

- Is it more efficient (cheaper/faster) to move the whole dataset to the cloud prior to computing over it, instead of repeated operations on smaller chunks, considering datasets coming into the cloud from external location?

- What is the cloud system inbound/outbound traffic and how can it be minimized?
- Is there static data that can be served by e. g. a geographically and cache optimized content delivery service, for lower communication latencies to external system actors?

3.3.6 Security

Security is a major concern in any information system. Although ADVENTURE is not a security project, it has created the architecture and its components with security aspects in mind and will cover a basic set of security aspects:

- Communication - Encrypted communication via a secure channel is a well-known pattern to secure data transfers on the border of a system. It should be possible with more or less effort, depending on the cloud type of choice to realize it in the cloud just the same as on a conventional system. Further, cloud providers offer the option to create VPNs and communicate in a safe and trusted environment.
- Data - A common approach at data security in the form of files is their encryption. This is highly specific to products and providers to cover in detail here.
- Application - General best practices for designing secure applications are applicable regardless of the cloud environment although the concrete implementation may vary depending on the choice of cloud type and vendor (and related constraints if any). An example general good practice is to never run software with a user with administrative privileges unless absolutely necessary.

IaaS providers let you specify rules to control the inbound traffic to the instance in a firewall-like manner or provide a firewall service or let you install one and manage it yourself. In effect, it is possible to completely reduce access to application layers nodes down to specific communication channels with expected request source addresses and open e.g. only https secure communication to end-users.

4 ADVENTURE Components Overview

As described in the previous sections the ADVENTURE framework consists of several components, which will be defined in the following subsections. Each section contains a description of the purpose, a diagram describing the internal architecture of the component with its subcomponents (called modules) and a first introduction of the functionality of the specific component. Further, the logical connections to other components are described. All connections to different components don't show the Message Routing if possible, as all communication is routed through this component and it is considered part of the architecture.

The technology foundations for the components are decided in later deliverables and therefore no direct technology implementation is mentioned in this document.

Most components contain an architecture diagram to illustrate the subcomponents or the connections within the component or connections with other components or systems. Figure 3 shows the meaning of the used symbols and stereotypes employed in the diagrams.

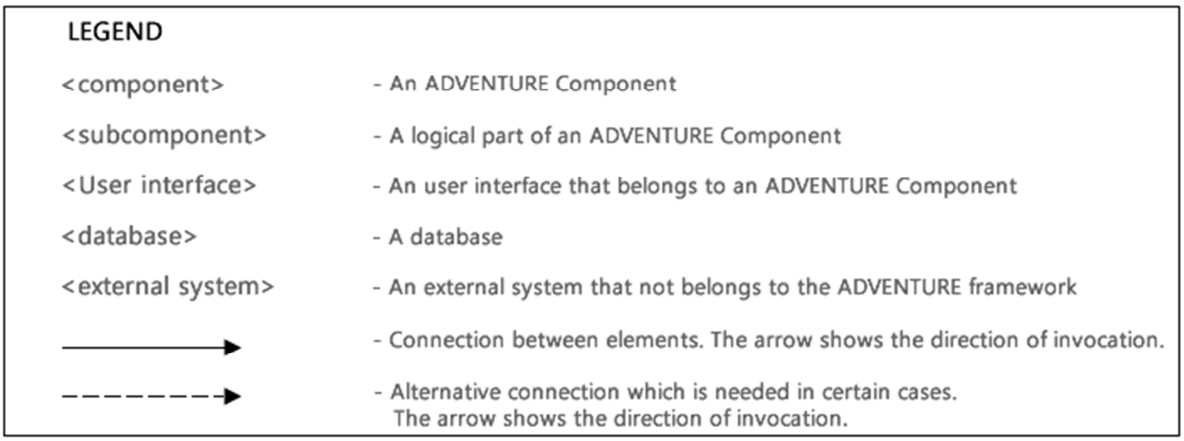

Figure 3: Component Architecture Diagram Legend

4.1 Dashboard

4.1.1 Component definition

The Dashboard will be the entry point for the users to the ADVENTURE Platform. From the Dashboard the users will be able to configure, monitor and view all related ADVENTURE data and processes for their company and Virtual Factories. Based on the login credentials used when logging into the ADVENTURE Dashboard, the role and privileges of the user can be determined, which will be stored in session data which is synchronized from the application server to the Cloud Storage component, as this is the security provider for the platform and needs this data for access control lists and enforcement of privileges. Figure 4 gives a schematic overview of the dashboard architecture.

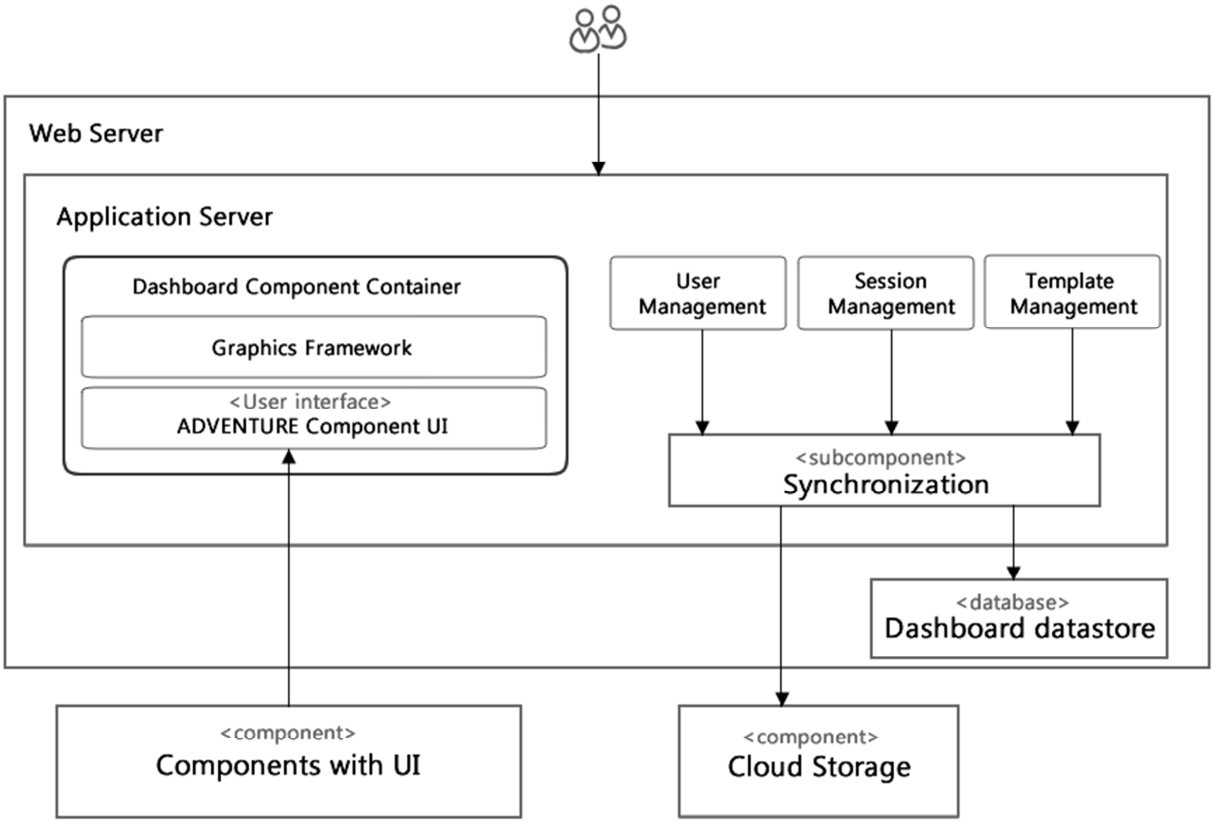

Figure 4: Dashboard Architecture Diagram

The core of the Dashboard runs in the application server and users access the UI this way. The application server is deployed on a web server.

The application server consists of the following components:

1. User Management – This component provides the UI to create, manage and delete user accounts. User management will ensure the authorization and authentication of users within the ADVENTURE platform. Based on the assigned privileges the user can view and manage the ADVENTURE processes and data based on his role and privileges. For this to work in ADVENTURE where user management is basically done in the Cloud Storage for other components than the Dashboard, this needs to be synchronized in with the Security Provider in Cloud Storage.

2. Session Management – This component will manage the user sessions within the ADVENTURE platform. This process will keep track of who is logged in, which part of the platform the user is viewing and what the state is of the running components. This functionality needs to be synchronized with the Security Provider in the Cloud Storages, as this is the security provider in ADVENTURE.

3. Template Management – The Dashboard will have a flexible UI to support the needs of different users. Not all users will be using the same components or functionality, as the Requirements Analysis showed that the Dashboard should

be configurable. Template Management will allow the user to create a role- or user-based view on the ADVENTURE Platform where necessary.

4. Synchronization – As already mentioned in User and Session Management, this information is usually managed by the application container and needs to either be directly loaded from and saved into Cloud Storage, or be synchronized with the Cloud Storage in another way.

5. Dashboard Component Container – The application server will provide a framework to develop and run individual separated components on. Most of the UI made for the Dashboard will be developed as part of the Dashboard task.

6. Graphics Framework – A graphic framework will be used to develop the user interface of the individual components. The selection of the framework will be decided in the technical specifications and is also related to the selection of the application server.

7. ADVENTURE Component UI – Several components in the ADVENTURE Platform will need a UI. The specific UI parts of these components could be separated from the component itself, but the technical implementation will be decided in the technical specification. The UI of this component will be bundled in a Dashboard container which could be used as a drag & drop component in the Dashboard. The Dashboard will comprise many of these components. For example, the Cloud Storage will run as a service in the ADVENTURE platform but the configuration of the Cloud Storage will be done in the Dashboard and for this purpose a Cloud Storage UI component can be developed which will be presented using the Dashboard.

4.1.2 Interaction of the component

The ADVENTURE Dashboard is the main UI component of the platform and thus has many logical links to other components:

- Message Routing
- Cloud Storage
- Simulation & Forecasting
- Execution
- Optimization
- Monitoring
- Process Designer

All physical communication with these components will be handled by the Message Routing component. Additionally, it might be necessary to have a UI for configuration tasks for all connected components, but this is intended to be kept as lightweight as possible.

The session information of logged-in (authenticated) users is usually managed by application servers that will be the basis for the Dashboard (see section 4.1.3), but as the session information needs to be available for the other components too, the Dashboard application server will be modified in a way that the session information is synchronized with or can be stored in the Cloud Storage component and can be used for all components including the Dashboard component.

4.1.3 Technical foundations

The core of the dashboard will be the application server. It is not the intention of this project to build a new user management application, portal system or application server from scratch. Therefore, an existing portal/collaboration platform will be used and adapted to the needs of ADVENTURE to host the Dashboard.

An existing UI framework needs to be selected to build the UI components for the dashboard. This framework needs to be able to run in the selected application server. The selected application server will need to be modified to the session information for authenticated users can be synchronized or directly stored in the Cloud Storage component. Alternatively, a single-sign on solution can be implemented, but these decisions will be provided in D3.2 and D3.3.

4.2 Cloud Storage

4.2.1 Component definition

This component is the central data storage of the ADVENTURE platform. It consists of a software system that receives requests or changes to data, selects a suitable cloud data store and executes the request on this store, sending the response back to the caller. Internally it integrates different storage types like binary data storage (non-queryable files), semi-structured data storage (like XML, JSON or other structured text files) or semantic data storage (highly structured text that allows semantic querying). For each internal storage type, it provides one external interface and contains one module to handle the storage concept (cf. Figure 5). If errors occur they will be forwarded to the requesting component, if they cannot be handled by the component itself (as not all problematic situations or events can be foreseen), then they can be handled by the software.

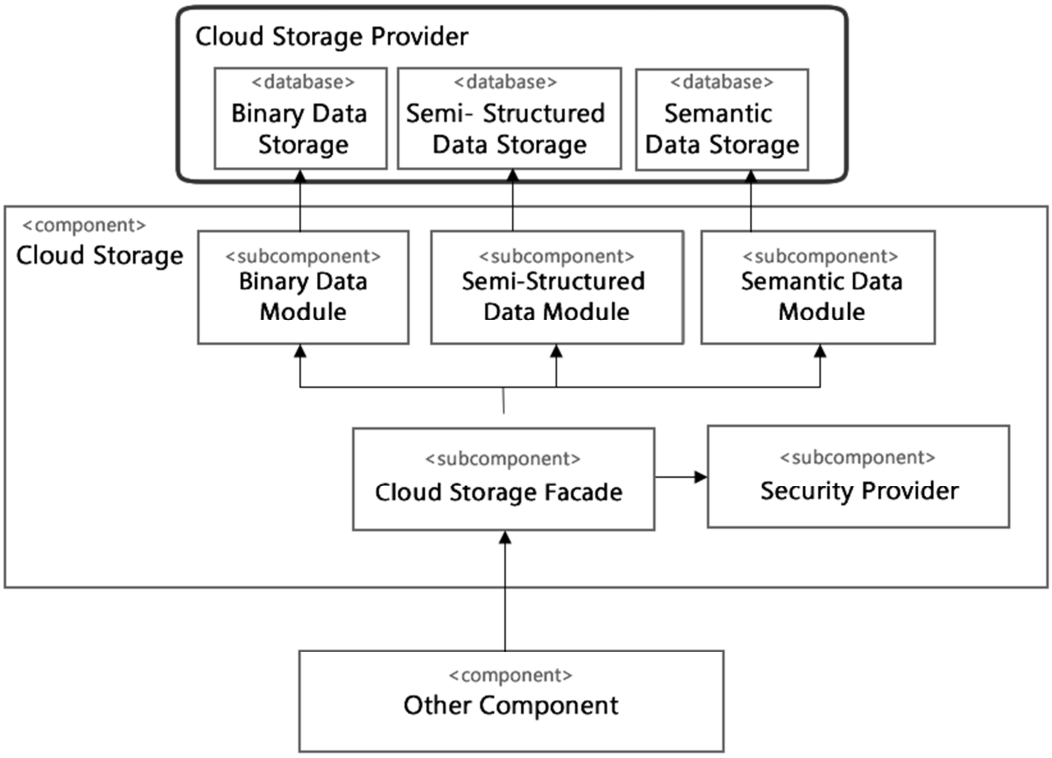

Figure 5: Cloud Storage Architecture Diagram

To increase a fail-safe availability (failover) and response time (elasticity), it is possible to run several instances of the Cloud Storage component in parallel. In this context it is necessary to guarantee data consistency. The load balancing should be provided by the Message Routing system (cf. section 5.) that will spread the load between separate instances of the Cloud Storage component.

4.2.1.1 Security Provider Role

For reasons of data security the connection from and to the Cloud Data Storage Engine has to be secure. Additionally the component has to check, if the requesting caller has the required rights to read or write the data. The security data will be stored in the Cloud Data Storage too, so that the security provider role can be fulfilled by the Cloud Storage component.

As Security Provider, Cloud Storage stores user accounts, organizations, virtual factories as well as access rights and roles with their respective connections. It will provide a separate interface to store the session context and request the rights and roles of this context that is otherwise managed in the Dashboard, to be able to identify logged-in users, components and systems. The data security system will be needed in the Cloud Storage and the Gateways to secure data in the cloud, in Legacy Systems and in Smart Objects so only authorized users and systems can access it.

Alternatively, the Cloud Storage can host the relevant data and be the provider for an Open ID solution that provides the users for all components with a single-sign-on solution. In any way, a way to identify a user by ID and assign security or role-

relevant data to the user is to be implemented. D3.2 (Functional Specification) deliverable will detail how the authorization process will work, and D3.2 (Technical Specification) will detail the implementation of the authorization and authentication details.

Access control to features and data can be based on association with a virtual factory, a company, a role or specific, fine-grained privileges.

4.2.2 Interaction of the component

Cloud Storage will only directly interact with the Message Routing component and the different cloud storage providers. Out of band messaging of files will resolve file transfers that directly connect sender and receiver as detailed for the Message Routing component. Therefore, the technology choice for the Message Routing will have to provide out of band file transfer. All ADVENTURE components will be able to interact with the Cloud Storage component (at least indirectly through the Message Routing component), because most components provide data to and depend on data from the cloud storage.

4.2.3 Technical foundations

The query languages used to obtain data from the Cloud Storage is based on the query languages of the cloud storage providers, because the queries will be forwarded from the Cloud Storage component to the different databases. Rewriting the different necessary query languages like SQL or SPARQL to be a unified query language limits the expressiveness for developers using the Cloud Storage and is outside the scope of ADVENTURE. Interfaces will be provided to access the conceptually different storage systems.

4.3 Data Provisioning & Discovery

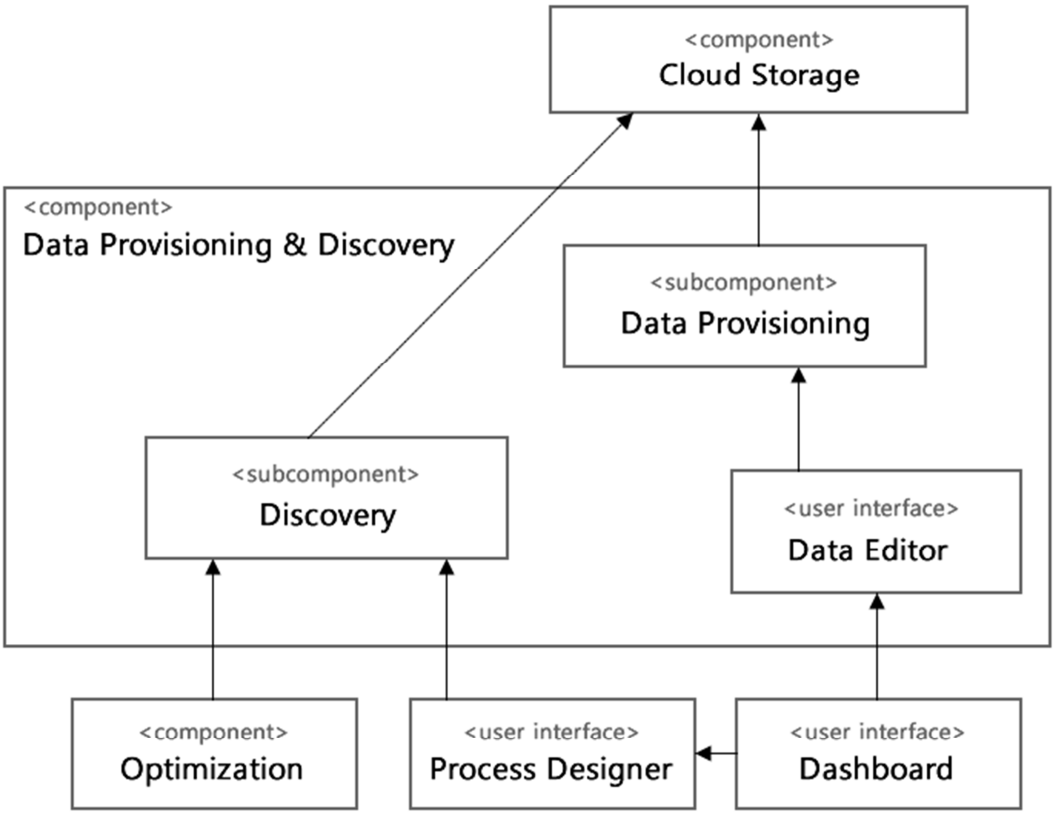

Figure 6: Architecture of the Data Provisioning & Discovery component

This section provides an overview of the architectural positioning of the *Data Provisioning and Discovery* component in the ADVENTURE system, the data model it uses, as well as some design considerations that need to be applied in the concrete functional and technical design.

4.3.1 Component definition

The *Data Provisioning and Discovery* (DPD) component provides a model and tools for description and discovery of key artefacts in the ADVENTURE system, required for the design, management and operation of Virtual Factories.

4.3.1.1 Data model

The DPD component defines a suitable data model in order to support efficiently ADVENTRUE tools and users in their information management activities.

In brief, the objects of data modelling are *partner entities* (clients, suppliers), *services* and *resources* that those objects provide in the manufacturing processes (i.e. shared among activities), and *processes* themselves.

- The *partner entity descriptions* specify ADVENTURE member organisations profiles. 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.
- *Resources* are often shared as inputs and outputs of manufacturing activities and sometimes even crossing the organizational boundaries. The *resources* are therefore also specified unambiguously through a unified data model that ensures semantic and syntactic compatibility across organizations and their services.
- *Process descriptions* support the processes search, the management of best practice process templates, sharing and automatic recommendations and process adaptation.

4.3.1.2 Key roles

The component may be further decomposed with respect to two key functionalities:

- Data provisioning
  The DPD data provisioning functionality will be realized by tools to model ADENTURE artefacts like organizations profiles, services provided by those organizations that can be orchestrated in collaborative processes, resources they consume and deliver, process templates, i.e. mainly static data, describing the overall business of an organization. The data provisioning will realize also the persistent storage contract of the provisioned artefacts and will store the structured data as result of modelling.
- Data discovery
  The DPD data discovery uses the model employed in the data provisioning to describe the respective artefacts. It allows for defining a criteria and then matching it against the provisioned in a persistent storage (by data provisioning) artefacts.

4.3.2 Interaction of the component

Figure 6 describes the dependencies between the DPD subcomponents and other ADVENTURE components or external agents.

The physical persistence of the provisioned data models is out of the scope of DPD module and depends on a persistency contract with an external component, whose role in ADVENTURE is performed by the Cloud Storage. In this dependency relationship DPD is the client that consumes, and the Cloud Storage is the service that provides the data persistency.

The consumers of the component services are agents and human users.

The agents are the Optimization component and the Process Designer. Both utilize the query capabilities of the DPD component to get eligible services, possible and preferred partners, etc. in the virtual factory lifecycle phases. The Process Designer

also uses the component to store process annotations and further provides assisted design to the users via the component's query interface.

The UI components integrated in the Dashboard can also utilize the query service interface to consume information for system assets. Following the same pattern third party applications can also be developed to repurpose the ADVENTURE information and create different scenarios using the data models.

In the provisioning phase (JOIN, see D2.1) the human users are provided with an intuitive UI in order to register new partners and a generic search interface for the different assets that the system maintains. The user interface is intended to be integrated in the ADVENTURE dashboard.

Other ADVENTURE components' user interfaces may utilize the DPD services and may create specialized views as appropriate for the scenarios they realize.

4.3.3 Technical foundations

The design goals for DPD component are to deliver software that takes full advantage of opportunities provided by cloud solutions for scalability and reliability. It should integrate seamlessly with the other ADVENTURE components, efficiently providing valuable services via well-defined contracts that allow various implementations to step in that role. The component will also define and manage a model with sufficient level of expressiveness and extensibility. The following paragraphs discuss design constraints and recommendations in these strategic directions.

Scalability and reliability

The scalability and reliability of the component is ensured by a design that is harmonized with the virtual infrastructure's scalability characteristics. It will account for reliable failover strategies, seamless bootstrapping of new and parallel processing nodes for horizontal scaling of autonomous, stateless processing algorithms as recommended in section 3.3 of this document.

The design will also imply efficient communication with the component, reducing wasted requests by detaching long running jobs, minimizing the network and processing overhead as well as lightweight architecture for the service frontends when appropriate.

Communication patterns

Aligned with the integration pattern in the system and aiming to achieve loose coupling, the component shall integrate with other components via the system's message service. The messages' content is structured in a standardized, open format (e.g. ATOM, JSON, XML, etc).

In potentially long running transactions, the component's communication strategy will be to utilize publish-subscribe pattern and push back response once ready (as opposed to alternatives to constant asynchronous query for ready state). This will reduce significantly the wasted network traffic, and will increase the overall performance and responsiveness of the system.

In addition to the message-oriented communication strategy that the component will employ, it will also implement service interfaces to offer direct communication lightweight REST Web APIs for traditional pull communication. This will guarantee not only the interoperability within the ADVENTURE system, but will also provide for other options for integration with third-party applications, that might repurpose the information supplied by ADVENTURE for example for Business Intelligence operations and other statistics.

Security & privacy

The component communicates potentially private resources between the user and the data store and this requires secure communication channels. These will be realized by the Message Routing component. DPD will provide service to authenticated users and agents and will delegate authorization to the Cloud Storage component, providing the user context, according to the authorization policy in the system.

The roles so far identified to be important to the component are: content owner, content consumer. Content owners are authorized to create, change the content they have introduced and to publish their content to selected, all, or no consumers. Content consumers are authorized by content owners to read their content.

Realizations of client applications that consume component services need to comply with the security constraints in the web infrastructure. For example, web clients that consume services remotely need to be deployed in the same domain to comply with the single origin request policy required by browser when using AJAX request (i.e. XHR). Alternatively, an appropriate strategy should be implemented weighting pros and cons concerning integration and security requirements in the system.

The component shall be secured according to best practices that mandate secure communications, reliable authentication of requesting users, etc.

Data model

The data model defined by the DPD component needs two distinct characteristics:

- *expressiveness* required to support the rest of the ADVENTRUE tools and users in their information queries,
- *extensibility* to facilitate the adoption of the ADVENTURE system and foster a large community (e.g. contribute to SEMIC.EU/Joinup or similar initiative of large scale).

The data model(s) is a structured and/or a semi-structured data format that allows for efficient, comprehensive queries targeting agents as well as human users.

It is out of scope of the data model employed by the component to span over manufacturing products and standards beyond the bare minimum but rather reuse and reference existing efforts in this area. It is in scope that this model can span over new products, services and standards that facilitate collaborative manufacturing and for that purpose the formalization of the model will use an adequately extensible and generic technology.

4.4 Message Routing

4.4.1 Component definition

The Message Routing component will take on a central role for the ADVENTURE Platform: it will keep a list of component instances, external systems and users of the platform and route messages and events from the transmitter to the receiver (see Figure 7). A holistic messaging protocol and message format is used between the ADVENTURE components and users. For the communication with external systems Gateways (see section 4.6) will be implemented which will transform the message formats and protocols as necessary for one external system each.

The logical connections that need to be established between the components and that need to be managed by the Message Routing component are visualized in Figure 8. All lines that cross the middle green layer “Data Exchange” will be handled by the Message Routing component.

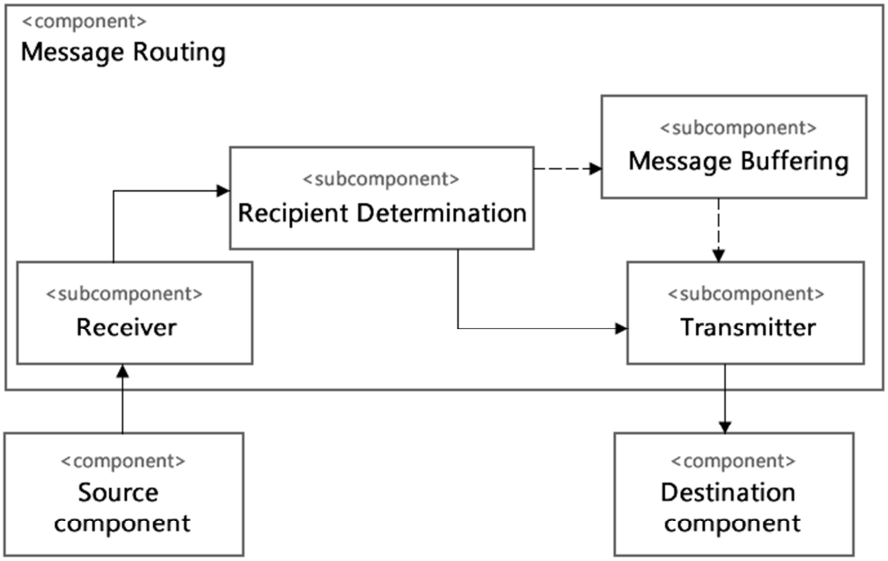

Figure 7: Architecture of the Message Routing component

Figure 8 also illustrates the need for a holistic messaging solution inside of ADVENTURE. Given this many components need to manage communications and keep up with the different variations of diverse components, a Message Bus-like solution is needed. As the components messaging is basically written from scratch for all communication with the rest of the ADVENTURE Platform, a shared format and protocol will ease the implementation and ensure that a defined standard in terms of security, reliability, scalability and message features can be enforced platform-wide.

To ensure an error-free communication the Message Routing component will guarantee the delivery of the message by providing message buffering, which stores the message until it can be delivered to the recipient, and providing an

acknowledgement feedback to the caller when the message has been delivered. Additionally a secure / encrypted connection will guarantee the necessary data security. Out of band communication for file transfers has to be enabled by the component, as file transfers otherwise can greatly decrease performance. The Message Routing component will also provide a real-time availability status for all message partners. To realize a high degree of performance, reliability, scalability and stability the Message Routing component should be 'cloud-enabled' and be able to run on several servers (see Section 4.1), which can communicate and provide fallback, which is extremely important to not provide a single point of failure for the entire platform.

Other components can use the Transformation Services component (see section 4.5) via the Message Routing to perform message transformation when communicating with external technology like smart objects, ERP systems or other legacy systems. The Message Routing itself can use the Transformation Services component to transform message formats, if a component does not use the holistic formats because of technology choices made later on. It should be noted that this is not expected and should be an exception as all components should adhere to the same holistic protocol and message format defined to internally communicate with other ADVENTURE components.

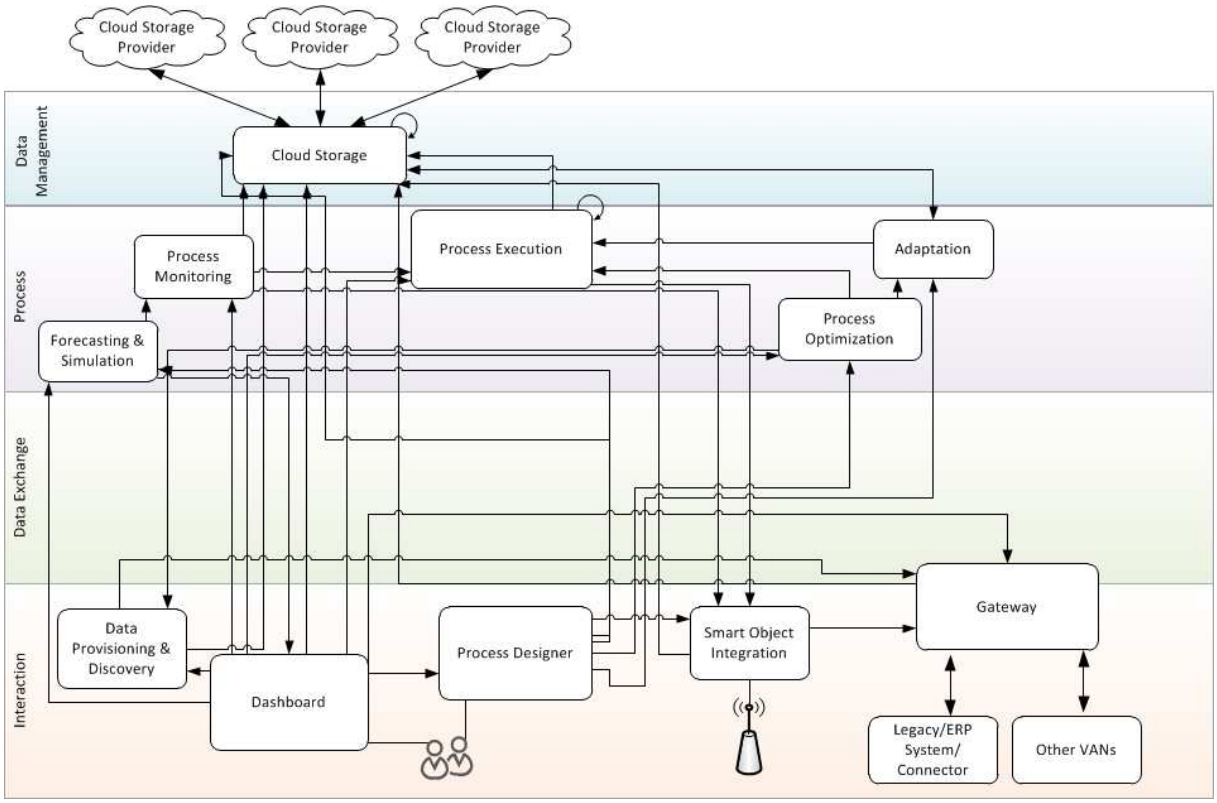

Figure 8: Logical Architecture of the ADVENTURE platform, demonstrating communication between the components necessary without Message Routing

4.4.2 Interaction of the component

The Message Routing component will directly interact with all components of the Platform. The component will define an appropriate message format and a holistic message protocol. The communication with external systems like Smart Objects or legacy systems will be realized via Gateways (See 4.6), which will translate the message formats and protocols. To achieve this they use the Message Transformation services (see section 4.5) that are services for Message Transformation. The Gateway components will handle Communication Protocol Translation as necessary for the external communication partner.

4.5 Transformation Service

4.5.1 Component definition

The Transformation Service is a complementary service to the ADVENTURE Platform. It performs pre-defined transformations from one message format to the other. These transformations could be needed when connecting external systems or VANs to the ADVENTURE Platform. The messages of the external systems might not be in the holistic ADVENTURE message format so they need to be transformed in order to be processed by other ADVENTURE components.

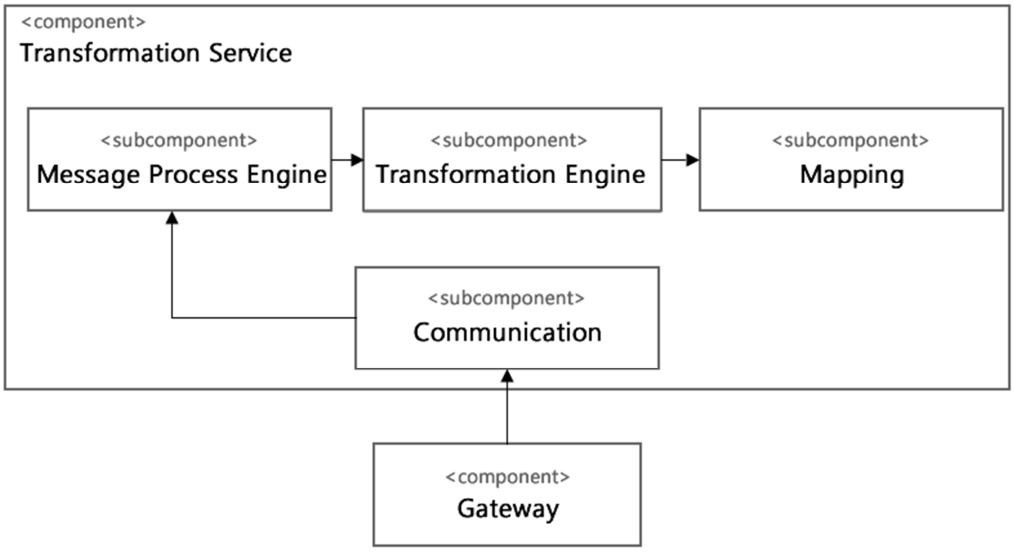

Figure 9: Transformation Service Architecture Diagram

The ADVENTURE Transformation Service consists of several sub-components (see Figure 9).

- The communication component manages all the messages that need to be received and send to the ADVENTURE Platform. It will take care of the wrapping of the messages and assigning the correct recipients.

- The Message Process Engine is responsible of processing the message to determine the sender of the message, what type of message it is and checks if there is a transformation mapping available for this message.
- If there is a transformation mapping available the message will be send to the transformation engine, which will use the available transformation mapping to transform the message to the required format. The transformed message will then be processed again by the process engine and be sent to the ADVENTURE Platform by the communication component.

4.5.2 Interaction of the component

The transformation component communicates with other components of ADVENTURE via the messaging component. If there are transformations to be applied to messages that cannot be sent by the Message Routing, these messages will be wrapped in a binary file and be sent as an out-of-band file transfer.

4.6 Gateways

4.6.1 Component definition

The ADVENTURE Gateways will be developed when needed for specific external systems. This could be Legacy/ERP systems, Value Added Networks (VAN) but also an external gateway running at an ADVENTURE member. Thus, some parts of the gateway are custom made as not all ERP or legacy systems share the same interface.

All the ADVENTURE Gateways share the same architecture and have a common interface to the ADVENTURE platform. Some sub-components are generic and will be used for every Gateway but each Gateway will require the implementation of a custom component to implement the specific interfaces to external systems.

The following figure identifies the sub-components used in the ADVENTURE Gateway and the external connections.

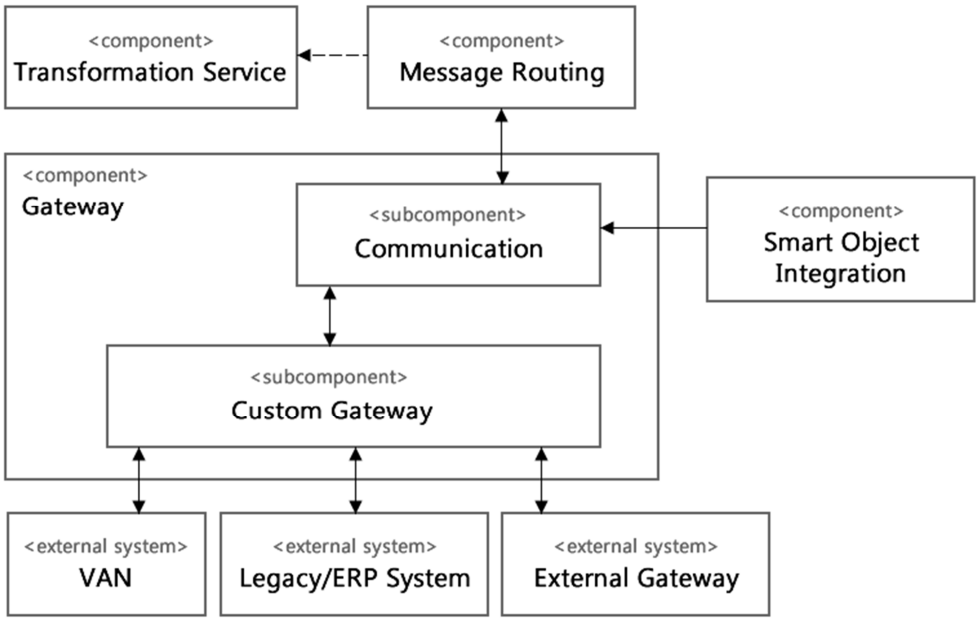

Figure 10: Gateway Architecture Diagram

Each Gateway will use the Communication sub-component. This sub-component is the bridge to the Message Routing component and will send and receive the messages on behalf of the Gateway. The component will use a common interface to communicate with the Custom Gateway component of the ADVENTURE Gateway. The Communication sub-component is also used as the connection point for the Smart Objects.

For each Legacy, ERP system or VAN a Custom Gateway component needs to be developed which will be the actual interface to the Legacy / ERP system. This could be a simple interface like picking up a file from the file system to a more complex interface like an RPC connection to a SAP system. The custom component will have a custom interface to the ERP / Legacy system and a fixed interface to the other sub-components of the Gateway component.

There will be a very simple Custom Gateway component called the “Default Gateway” that can connect to an External Connector component. This External Connector is an external system that meets the requirements of the Message Routing components message format and protocol, so that in theory could directly connect to the ADVENTURE Platform via the Message Routing. For security reasons, this lightweight Default Gateway component will forward the messaging between externally developed connector components and the Message Routing.

4.6.2 Interaction of the component

The component will interact with external systems like ERP systems or existing Gateways running at the ADVENTURE member’s location. This could be as simple as picking up a file from the file system but also more complex connections, for example SOAP, database and Client-Server connections.

Messages from the Smart Objects are also handled by the Gateway and the Gateway is the interface to ADVENTURE Platform for these Smart Objects. The reasoning for this is that Smart Objects will likely also have diverse interfaces and therefore would need the same kind of Gateway for each different type of Smart Object. Additionally, Smart Objects could be integrated into a factory of an ADVENTURE member, or belong to the controlling company which might need special cases necessary that would need a special implementation.

Messages received from external systems will be forwarded to the ADVENTURE Platform via the ADVENTURE Message Interface and will follow the defined message format and communication interface. The ADVENTURE Message routing will make sure the message will be send to the correct recipient.

4.7 Process Designer

4.7.1 Component definition

The Process Designer component realizes the use cases for interactive Smart Process formalization into format that is executable by the Process Execution Engine and reusable by the Process Designer itself. It provides companies with the functionalities needed in order to define, design and orchestrate manufacturing processes, comprising production steps, partner services and manufacturing resources. The standard process design routines will be further enhanced by adventure tools and services for recommendations for suppliers and favourites list, as well as process and activities annotation assistance. In addition, the Process Designer shall integrate with the Simulation module to get design support by means of computing risks and key performance indicators with different partner assignments and constraints. Those functional dependencies define the dependencies to the components that provide them. The diagram below summarizes that.

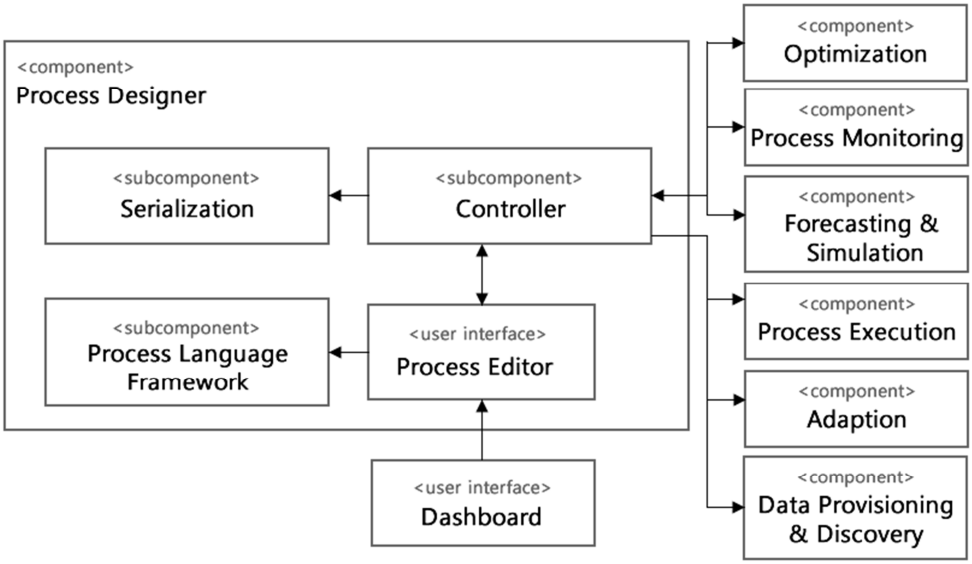

Figure 11: Process Designer Architecture Diagram

The use case to give users the ability to create and modify Smart Processes is realized by means of a User Interface application, integrated by the *Dashboard*. To realize the required functionality, the module depends on:

- Data Discovery and Provisioning component to store and retrieve process-related metadata and perform search and recommendations based on it.
- An Execution subcomponent to deploy designed processes.
- Optimization component to request recommendations for changes in process structure and activity bindings (partners assignments).
- Simulation to analyse processes for assessment of risks and impact on selected criteria.
- Cloud Storage to find user preferences, partner factories processes, favourite lists and other user related data.

4.7.2 Communication patterns

The module shall employ best practices and patterns for efficient communication with remote services that will not affect negatively the UI responsiveness on one hand and on the other will minimize the network utilization and the waste load on the services.

4.8 Process Execution

4.8.1 Component definition

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

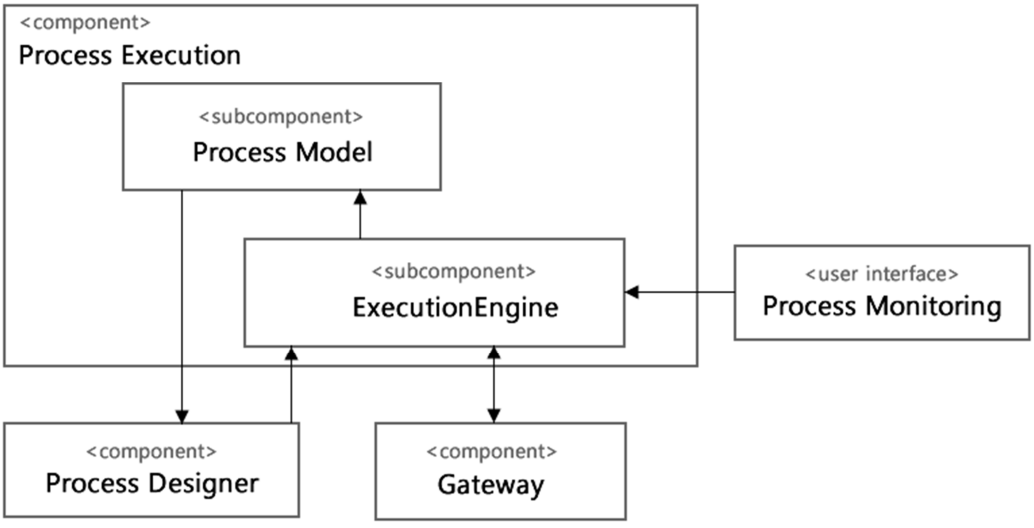

Figure 12: Process Execution Architecture Diagram

Smart Processes typically consist of

- Tasks: calls to operations in the Gateway services of virtual factory stakeholders.
- Data Elements: the output of operations calls, which can serve either as input for other calls or represent the result of an execution.

ADVENTURE will be purely process driven. To further exemplify this statement we will give the following example processes:

- Order process: the interaction between several partners to handle an order. This includes sequential or parallel calls of operations in Gateway web services of multiple stakeholders in a virtual factory. The output of each operation can serve as the input for a subsequent operation, with potential transformations as part of the process. *This process is long-running and potentially involves human interaction on behalf of stakeholders. This kind of process is referred to as a macro-flow.*
- Reading stock level: reading the stock level of a stakeholder in a virtual factory may either be as simple as a one step process, that calls an single operation of a partners Gateway, or it may consist of a series of calls, one for each product or good provided by the stakeholder. In either case the result will exist as a data element, which represents the result of the process. This process is short-running (goal <10 sec), and involves no human interaction on behalf of process stakeholders. *This kind of short-running process with no user-interaction will be referred to as micro-flow.*
- Reading values from smart objects: accessing data from smart objects is handled entirely by the Gateway implementation of the owning stakeholder. This can decide to use stored values (usually stored in Cloud Storage) or directly call the smart object for real-time data. Usually this is supposed to be a micro-flow. As there might be the possibility that the Gateway doesn't answer quickly, a solution for this is necessary, e.g. using non-realtime-data from Cloud Storage.

The Process Execution component will handle both in the same way, micro-flows and macro-flows. The difference between these is that the Dashboard as a user-interface to Process Execution will either allow the monitoring of macro-flows, or the immediate display of results from a micro-flow in a minimal form.

4.8.2 Interaction of the component

Users will not directly interact with this component but instead utilize the dashboard to invoke, adapt and monitor process.. The Process Execution component will have to provide the following minimal interface:

1. Create new instances from a process model through a yet to determined standard language (e.g. BPEL, BPMN).
2. Starting process instances.
3. Stopping running process instances.
4. Adding new data elements for running process instances.
5. Adapt the process model of an instance, including the values of data elements. The adaptation of steps includes replacing, reordering and adding new steps. It has to be noted that it will be necessary to add new gateway service

(representing drop-in virtual factory stakeholders) on the fly, which is not a common feature for most BPEL based engines.

6. Restart adapted process instances including skipping steps, or redoing already executed steps.

Further interface properties as required by monitoring, simulation and forecast will be specified in the respective sections.

4.8.3 Technical foundations

This section is a mere recommendation to the tasks T3.2 and T3.3 and will also be provided in the separate document that has been created alongside this deliverable to take the functional and technical remarks that were made during the work for D3.1 to the following tasks.

In conjunction with the requirement to have every call to an operation in a Gateway service be handled through the Process Execution component, the Process Execution component should be implemented with the following properties:

- Replaceable: The well defined interface and protocol of the component, as well as its compliance with the state of the art, shall allow for transparent exchange of different implementations that require reasonable effort to implement.
- Service Oriented: The consortium strives to provide the engine itself as a service (REST or SOAP), to ensure property 1. This includes a standard interface and event model that every engine has to implement.
- Service Level Modularity: Monitoring, adaptation, simulation, optimization will NOT be part of the engine. Any engine plugged into ADVENTURE should be able to use these components, which will be also provided as a service, as a first class citizen.
- Standard Event Model: In order to allow for decoupled monitoring, adaptation, simulation the engine has to support a standard event model as proposed by Mangler et al. (J. Mangler and S. Rinderle-Ma, “Rule-Based Synchronization of Process Activities,” in 2011 IEEE 13th Conference on Commerce and Enterprise Computing (CEC), 2011, pp. 121–128.).

4.9 Process Monitoring

4.9.1 Component definition

The Process Monitoring component will be an independent service that relies on the events provided by the Process Execution component and indirectly by the Dashboard and by external systems via the Gateways and Smart Objects.

The different events may be outlines as follows:

- Events from the Dashboard and Process Designer will include all activities conducted by ADVENTURE brokers regarding the modelling and optimization of processes.

- Events from the Process Execution component will include all process execution related aspects. Processes are the hubs that control the invocation of all other components (including Gateways and Transformation services), as well as Simulation & Forecasting.
- Events generated by smart objects can trigger an own process in the Process Execution component. As such, events from other systems (e.g. Gateways or Smart Objects) will be handled via the Execution Engine, which means that the Monitoring will be connected to the Execution Engine from a technical perspective, although it will handle different events from different systems from a logical perspective.

As all components are either visually integrated into the Dashboard, or controlled through processes, monitoring can solely rely on a connection to these two components. All monitored information is stored in the Cloud Storage. The Dashboard will be able to subscribe to live monitoring events, in order to provide users with visual progress.

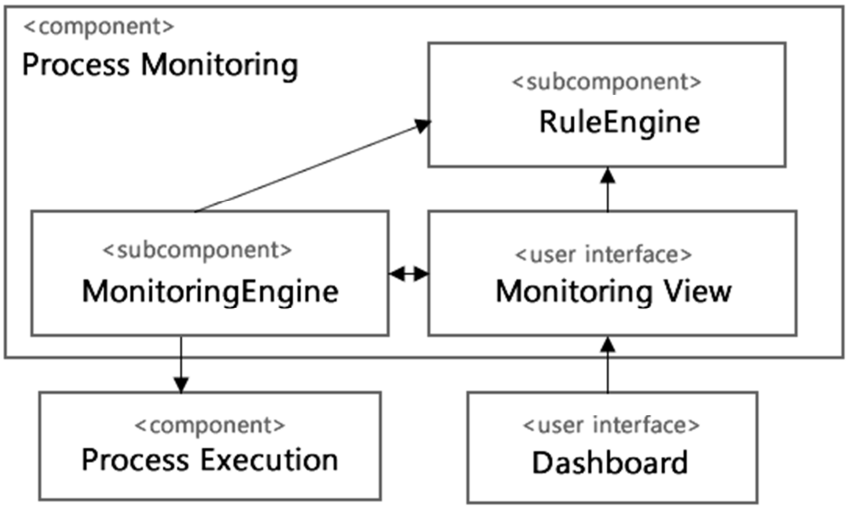

Figure 13: Process Monitoring Architecture Diagram

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

- Trigger the creation of new process instances, e.g. to write a message that updates an ERP system.
- 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.

So far, the following events from the Process Execution component were identified that the Process Monitoring component will rely on:

1. Task is about to be executed
2. Gateway service called
3. Gateway service call finished
4. Task is about to finish
5. Task has failed
6. Data element was modified
7. Process model was changed
8. Process model has an error
9. Partner was changed
10. Invalid Partner
11. Process has been stopped / started / finished
12. Process is stopping / finishing (intermediate state)
13. Task execution progress (with task specific progress report format)

4.9.2 Interaction of the component

The user will not directly interact with this component but instead use the Dashboard to consume alerts, notifications or process progress. The Monitoring Component will provide a subscribing interface that forwards event streams from the process component, in order to allow for visualization off process progress in the Dashboard. Besides from this stream, the component:

1. Pushes log data to the Cloud Storage.
2. Pushes notifications to the Dashboard.
3. Stops processes and pushes alerts to Dashboard.
4. Triggers new processes.

4.10 Forecasting & Simulation

4.10.1 Component definition

The Forecasting & Simulation component will be an independent service that relies on the Process Execution component. During a forecast the Process Execution is instructed to execute a process in forecast mode. E.g. when an order is executed in forecast mode, the Gateway service does not trigger the actual order, but instead returns information regarding various execution parameters connected to the invocation of a particular task. Examples:

- The predicted minimum, maximum and average time a task will need.
- The predicted output from a task. As a task may return arbitrary information (as defined by the Gateway), which may include costs, product quality, transformation results, etc.
- The set of possible outputs from a task, including the probability for each case. This set can be used to create an entire set of forecasts, with aggregated probabilities.

Forecasting will allow an ADVENTURE broker to assess the behaviour of virtual factory stakeholders, when involved in particular processes.

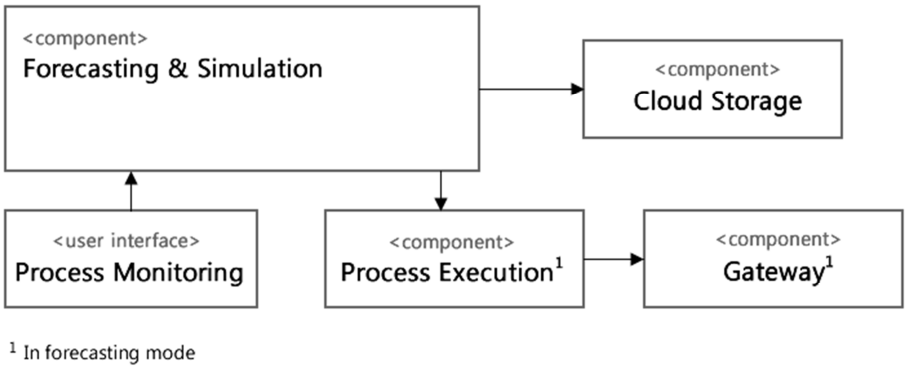

Figure 14: Forecasting & Simulation Architecture Diagram

The visualization of forecasting should take place in the same UI that will inform users about the state of execution, which is the visualization provided by the Monitoring component. This visualization should be clearly distinguishable from the current state and past state of execution, e.g. by showing forecast steps in a transparent way or another colour.

Simulation is the execution of an entire process in forecast mode, which can be visualized in the Dashboard through the Process Monitoring component, which must in turn understand that the simulated data is not to be confused with live data.

4.10.2 Interaction of the component

This component is intended to trigger the Process Execution component and monitor and collect (monitor) the execution of processes in forecasting mode. All forecasts will be saved in the cloud storage in a standard format that will be closely related to the process format. A forecast will be triggered from the Dashboard by an ADVENTURE Broker.

The information generated by the Forecasting & Simulation component is saved in the Cloud Storage and can serve as input for the Optimization Component if forecast data is needed.

4.11 Optimization

4.11.1 Component definition

The Optimization component offers a means to get an optimized proposition on which factories should be involved and assigned to which (abstract) process steps of a Smart Process. For different subsets of partners, different goal functions and constraints it provides a set of *execution plans* indicating which task of the Smart Process should be assigned to which factory. I.e. optimization solutions constitute execution plans and process configurations. It will be an independent service that proposes process variations strictly on a mathematical basis.

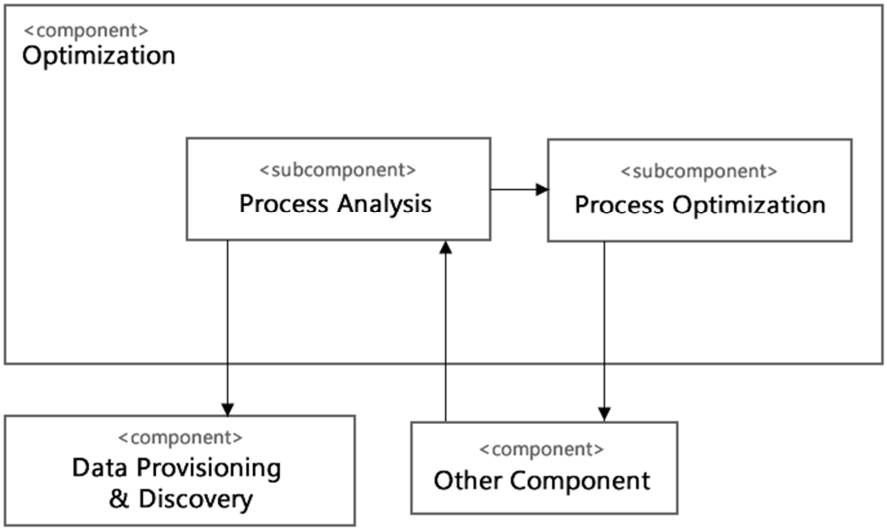

Figure 15: Optimization Architecture Diagram

For particular process models, with arbitrary process steps (each possibly referring to different virtual factory stakeholders) and arbitrary data elements it will for example operate with the following non-functional aspects:

- A set of partners that yields minimum execution time.
- A set of partners that yields lowest price.
- A set of partners that yields highest quality.
- A set of partners that produces something with the lowest carbon footprint.

Given a set of partners, tasks, restrictions and constraints regarding non-functional aspects of the overall process, the Optimization component computes an optimized assignment of partners to each task.. As the proposed optimized solutions depend on the data elements available in a particular process instance, e.g. price and quality, they can only be considered if they are actually part of the process. The base data for optimization will stem from two sources:

- For individual process steps: forecasts and historic process logs from monitored, past executions.
- Static partner information: entered by the Active and Passive ADVENTURE Members during the JOIN phase (for JOIN phase, see D2.1).

As forecasts are always intended to deliver timing information, calculating the minimal execution time, critical paths and slack will always be possible. Predictions (including ranges) for data elements (like quality or price), which are also delivered by forecasts, will allow for more elaborate suggestions.

A second possibility to deduce possible optimization solutions is from process mining: ADVENTURE brokers may continually adapt processes to include certain steps. The idea of process mining is to find out that a certain step is very often inserted into the standard model, and thus may be a candidate for an updated (optimized) process model. This type of optimization is therefore a widely automated task (in terms of assistive suggestions or complete automatic change). It is sensible for this approach to use e.g. data discovery to suggest for example best practice process sequences or corporate-approved sequences, etc. to change structurally the process for optimal effect and compliance

Optimization solutions are always to be stored in the Cloud Storage and can be the basis for process adaptation. The Optimization component will allow an ADVENTURE broker to generate a set of process configurations that are optimized regarding a particular set of properties, stemming from forecasts and partner properties. The final decision whether an optimized solution is selected, which version is selected, and how it is applied (i.e. valid for future processes or applied to current processes) lies always with the ADVENTURE Broker.

4.11.2 Interaction of the component

The Optimization component is to be triggered through the Dashboard and yields a set of process configurations. These configurations are to be available in the standard XML format used by the Process Designer, in order for an ADVENTURE Broker to explore. The Process Designer has to contain a means to select (for a particular process) data elements, a goal function and constraints. For process mining the Optimization component can use process logs from the Cloud Storage.

4.11.3 Technical foundations

Optimized solutions will be saved in the standard XML format that is utilized by the Process Execution component and Process Designer. Based on the structure of the Smart Process provided by the facilitator, the Process Optimization component initially models a linear optimization program for the selection of partners / factories. In order to compute an optimal execution plan, standard linear programming tools can be used, e.g., the commercial linear programming solver CPLEX[1] or the open source linear programming solver lp_solve[2]. As the computation of an optimal execution plan potentially requires strong computational efforts – depending on the problem size – a proprietary, heuristic algorithm providing “close-to optimal” execution plans can be applied. For process mining, a set of standard algorithms and tools [3] will be reused.

4.12 Adaptation

4.12.1 Component definition

The Adaptation component will be an independent service with the purpose to handle all runtime aspects regarding the adaptation of process models. Proposed adaptations may either be:

- elaborated by an ADVENTURE broker through the process designer, or
- proposed from the optimization as described in subsection 4.11.

[1] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
[2] http://lpsolve.sourceforge.net/
[3] http://www.processmining.org/

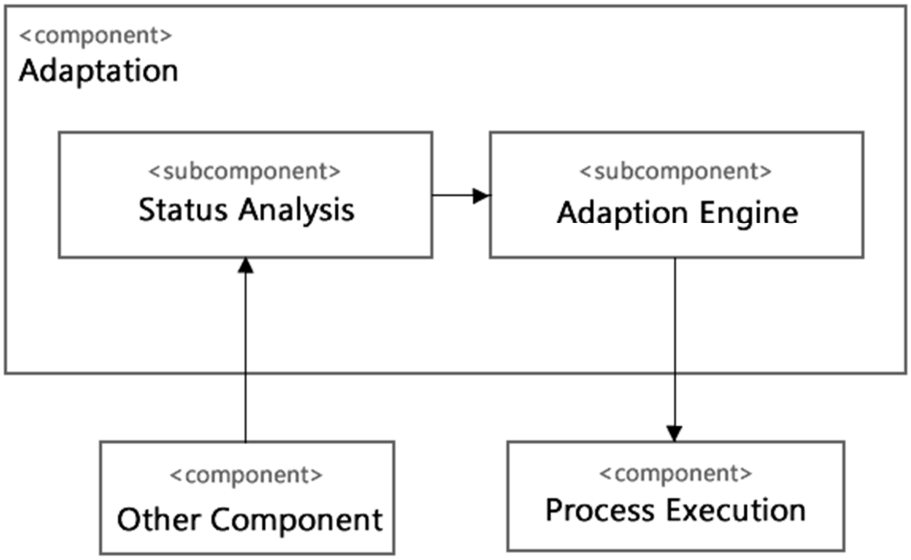

Figure 16: Adaptation Architecture Diagram

The main purpose of the Adaptation component is to deal with the application of optimization results to running instances. The first case, an ADVENTURE Broker selecting an optimized process model and using it to create a new instance, is trivial. However when such an optimized solution has to be applied to a multitude of already running instances, a refined approach has to be taken.

Thus an Adaptation component provides the following interaction possibilities, in conjunction with the dashboard:

- For a particular process propose a set of processes stemming from subsequent optimization runs or from the process designer.
- Instantiate a particular proposed optimized solution.
- Apply a proposed optimized solution to a set of process instances.

4.12.2 Interaction of the component

The Adaptation component reads optimized solutions from the Cloud Storage. It does not save any data and usually does not interact with other components, unless itis triggered by them or interacts with the Process Execution.

4.12.3 Technical foundations

Optimized solutions will be represented in the standard format of the Process Execution component and the Process Designer. This format will be defined and elaborated within the functional and technical specification of ADVENTURE (i.e. D3.2 and D3.3).

The Adaptation component will analyse the process model of particular instances, stop process instances, modify, and then restart them. Thus it will strongly rely on the properties given in Subsection 4.8.3.

4.13 Smart Object Integration

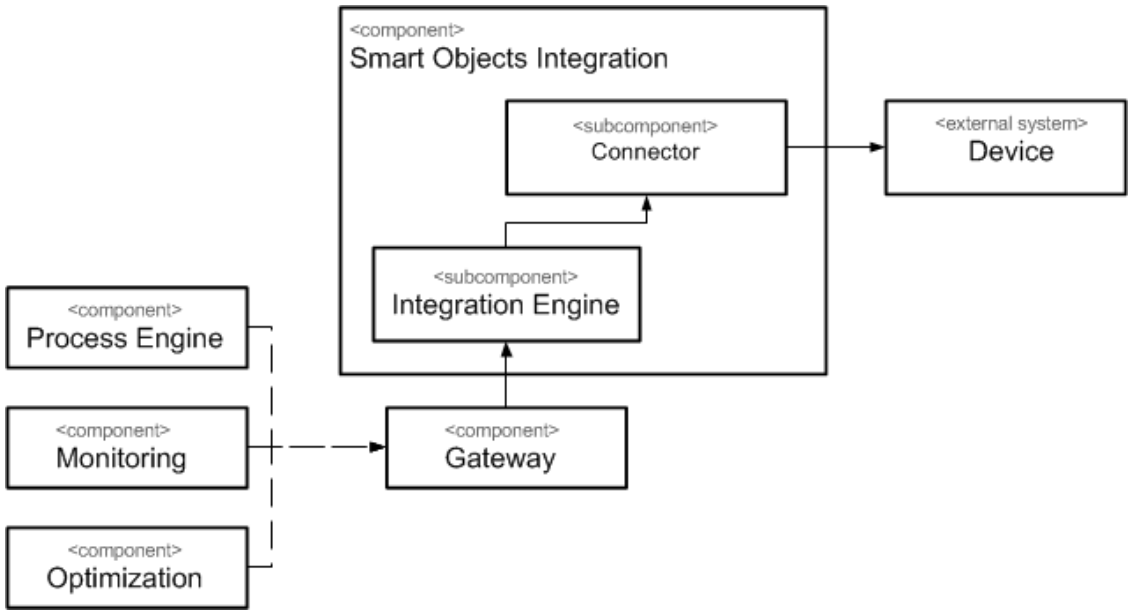

Figure 17: Smart Object Integration Architecture Diagram

This section provides an overview of the architectural positioning of the *Smart Object Integration* (SOI) component in the ADVENTURE system, the data model it uses, as well as some design considerations that need to be applied in the concrete functional and technical design.

4.13.1 Component definition

The *Smart Objects Integration* component has the role to seamlessly integrate physical devices (e.g. WSN[4], smart devices, etc.) as IT services in the cross-organizational collaborative processes, which is one of the key differentiators for the project. In the project Glossary such type of service providers are referred to as *Smart Objects*.

SOI standardizes the handling of *Smart Objects* in the ADVENTURE system, together with other components, in compliance with the defined lifecycle phases to seamlessly *provide* and *discover*, *compose* in processes and *interact* with them at runtime in ADVENTURE scenarios, abstracting from concrete operational and technical details of the backend devices.

The *Smart Objects* integrated by SOI in processes will deliver real-time information, important to the *monitoring* of the operational state of the process or facilitate decisions on their control flow or *optimization*.

[4] Wireless Sensor Networks

To enable this, the SOI tools and framework will be designed for the following key responsibilities:

- Make transparent all possible obstacles that a constrained device application might impose as compared to full scale services. Examples are reliability and short-lived lifecycles (i.e. sleep and be available only when needed).
- Implement efficient communication of data, which might include pre-processing and aggregations, optimized communication to backend and others;
- Take care how security and privacy concerns for integration of *Smart Objects* in cross-organizational scenarios are addressed.

4.13.2 Interaction of the component

The diagram on Figure 17 describes the dependencies between the SOI subcomponents and other ADVENTURE components or external agents.

The component is decomposed in two logical subcomponents to carry out its responsibilities:

- *Connector* to realize the connectivity with the physical devices and provide an API for interaction with them.
- *Integration Engine* to implement data provisioning efficiency and compliance mechanisms, security and privacy.

The consumers of the component services are agents – the other ADVENTURE components that need interaction with Smart Devices. Such components are *Process Execution*, *Process Monitoring*, *and Optimization*. The component interaction with the other components is outlined in section *4.6 Gateways*, following the standard communication patterns envisioned in this architecture document.

The subcomponent *Connector* processes data provided by external agents – the backend *devices* (e.g. in a Wireless Sensors Network or Smart Devices) and in special cases may invoke operations on them as well. The *Connector* is responsible to provide a reliable data transfer from the devices to the component and delegate further processing for optimization and compliance to the *Integration Engine*. The *Integration Engine* mediates the *Gateway-to-Device* communication. The *Gateway* itself is a thin layer that integrates the component in the communication infrastructure with the other ADVENTURE tools.

The composition of *Smart Objects* in collaborative processes and design of control flow decisions based on their output is carried out in the *Process Designer* component specified in section 4.7.

The provisioning and discovery of *Smart Objects* resembles the process for any other partner service and is supported by the model and tools defined in the *Data Provisioning and Discovery* component, specified in section 4.3.

4.13.3 Technical foundations

The simplest and widely deployed architecture for integration of devices in distributed software systems is to use an application gateway that implements all details around the interaction with the device via its specific protocol and the service enablement.

In the last years, the advances in the IoT technology area and more recently the move to the 'Web of Things' concept (similar to the 'Web of Data' concept) opened up for different perspective that allows for a more seamless integration of physical devices as information sources with the Web without necessarily using application gateways.

The focus of SOI is to provide a thin bridge that harmonizes and abstracts ADVENTURE from either of these approaches.

The *Connector* subcomponent can be an existing solution that implements connectivity to devices with some industry or other protocol and ADVENTURE shall explore the opportunity to reuse such solutions. On the other hand the project shall try to use different types of *Connectors* in order to prove the concept of seamless plug'n'play of devices in ADVENTURE. This may require the implementation of new Connector types as well.

The technical design of the subcomponents shall account for the unreliable and constrained nature of the devices and their environment and provide the necessary solutions to deliver information reliably and deterministically.

5 Potential Risks related to the ADVENTURE Architecture

As a matter of fact, all architectures have certain risks which are derived from their structures and from the way that the components are designed or connected. Since ADVENTURE is a complex project with many different aspects, the architecture has been widely discussed any analysed by the consortium. Nevertheless, there are certain risks which will be highlighted in this section. The consortium is fully aware of those risks and will carefully monitor them in the course of WP3 and WP1. The following sections show those risks and they also describe how the consortium handles them during the project.

The main risks were collaboratively identified during consortium discussions. The most important ones are system overhead, trust issues and reliability aspects.

5.1 Risk I: System Overhead

In other projects the consortium has experienced that superfluous features of complex systems often are accompanied by a big work overhead in terms of configuration and setup time, even when only a small subset of features of a broad solution are used for the targeted systems. This added workload can result in a lower overall quality of the technical implementation and therefore this overhead has to be minimized if possible by not planning system features to expand beyond the needed dimensions.

In past projects, superfluous system overhead was most likely to be found in the

- projects internal messaging infrastructure,
- oversized and unsuitable storage components and
- in complicated or inappropriate user interfaces.

In order to handle those risks, the consortium has discussed several principles for the functional and technical specification:

- The messaging component should implement a lightweight approach using a a shared messaging format and messaging protocol amongst all ADVENTURE components. Obviously, external systems that will need to interact with ADVENTURE (e.g. ERP systems or other value added networks) will not follow those ADVENTURE protocols in most cases. Therefore, the Gateways and the transformation services have been added as components to the ADVENTURE architecture. Those will handle all third party interaction with ADVENTURE.
- The storage component will not try to solve a storage for all types of data by itself. Instead, it will implement a façade pattern and split storage and query activities into three types: structured data, semi-structured data and binary data. This will allow the storage to reuse well-proven storage providers, which will be selected during the functional and technical specifications.
- The user interface will be driven by a holistic UI concept, to be implemented by the Dashboard. This concept will provide guidelines as well as an overall frame for integrating UI elements from the different components.

5.2 Risk II: Reliability Aspects

While easier to realize, monolithic architectures and components tend to be inflexible and provide single points of failure for the whole system.

Therefore all essential components e.g. the Message Routing, the Message Transformation, the Process Execution, the Dashboard and the Cloud Storage components should be redundantly deployable on cloud instances, which lead to high flexibility and reliability. An added benefit of this is a high scalability in case of high system load. This risk is mostly perceived for the Dashboard, Message Routing and Process Execution components as they are harder to replicate than other components. This aspect will be discussed in more detail within the technical specification of ADVENTURE.

5.3 Risk III: Trust Issues

ADVENTURE will provide an advanced approach for process execution and monitoring and it will allow replacing partners within a process in an ad-hoc manner. This will allow ADVENTURE to realize the concept of the “Plug & Play Virtual Factory". However, this also means that ADVENTURE needs to have access to certain company data such as the company profile or the process status. The architecture therefore foresees an own cloud based data storage and an own process execution engine. This will obviously require a certain level or trust by ADVENTURE users as they have to publish some of their company data. The consortium is fully aware that this may be seen critical by users but the consortium sees it as a necessary in order to be able to realize the ADVENTURE idea. It is therefore planned to be fully transparent to users by showing them what type of data is stored and why it is needed. Additionally, state-of-the-art concepts will be used for handling security and privacy aspects within the messaging and storage component and within the Dashboard.

6 Conclusion

This deliverable has defined the global architecture of ADVENTURE in a high-level way. This global architecture has been used to define components and their interconnection. It is based on the main components defined in the description of work but will be performed at a more detailed level. As such, some components have been detailed or even split into several sub-components.

After agreeing on the global architectures definition and having first detailed definitions about how the components are going to work and interact, details will now have to be defined in the functional specification (D3.2). The technical implementation will have to be specified in D3.3. If any of the following deliverables will force a revisit of the architecture, which is not expected, the changes will be recorded in an extra deliverable.