Large Scale Integrating Project

Grant Agreement no.: 257899

D2.3 – SMART VORTEX Suite Architecture and technical specification

SMART VORTEX –WP2-D2.3

Project Number	FP7-ICT-257899
Due Date	2011-09-30
Actual Date	2011-09-29
Document Author/s:	Mathias Johanson, Lars-Åke Johansson, Jordan Janeiro Lopes Da Silva, Massimo Mecella, Tore Risch, Jens Grabarske, Dominic Heutelbeck, John Lindström, Dirk Schnelle-Walka
Version:	1.1
Dissemination level	PU
Status	Final
Contributing Sub-project and Workpackage	WP2
Document approved by	RTDC

Co-funded by the European Union

Document Version Control			
Version	**Date**	**Change Made (and if appropriate reason for change)**	**Initials of Commentator(s) or Author(s)**
0.1	110716	First version	MJ
0.2	110719	Tiered architecture	MJ
0.3	110915	DSMS architecture	MJ / TR
0.4	110920	Data capture, collaboration architecture	MJ
0.5	110921	Visual Query architecture	MM
0.6	110922	Refined structure, minor fixes	MJ / JL
0.7	110928	Tiered arch. overview, intro, collab. sect	MJ / JJ
0.8	110929	Rules and proc, sec 4. Minor fixes.	TR / MJ / JL
0.9	110929	Soc. Net., sec 6 added, minor fixes	JG / DH / MJ
1.0	110929	Final version.	MJ
1.1	110930	Sec 5 revised	DS

Document Change Commentator or Author		
Author Initials	**Name of Author**	**Institution**
MJ	Mathias Johanson	Alkit Communications
LÅJ	Lars-Åke Johansson	Alkit Communications
TR	Tore Risch	Uppsala University
MM	Massimo Mecella	Sapienza
JL	John Lindström	Luleå University of Technology
JJ	Jordan Janeiro Lopes Da Silva	Delft University of Technology
JG	Jens Grabarske	Forschungsinstitut für Telekommunikation
DH	Dominic Heutelbeck	Forschungsinstitut für Telekommunikation
DS	Dirk Schnelle-Walka	TU Darmstadt

Document Quality Control			
Version QA	**Date**	**Comments (and if appropriate reason for change)**	**Initials of QA Person**
V 1.1	30/09/2011		IK

Catalogue Entry

Title	SMART VORTEX Suite Architecture and technical specification
Creators	Johanson et al.
Subject	Software Architecture
Description	Describes the architecture of the Smart Vortex software suite
Publisher	Smart Vortex Project
Contributor	
Date	2011-09-29
ISBN	
Type	
Format	
Language	English
Rights	

Citation Guidelines

EXECUTIVE SUMMARY

This document describes the software architecture of the Smart Vortex suite. The Smart Vortex suite consists of a number of software components for capturing, processing and visually querying high volume data streams, together with collaboration and decision making support tools, including social networking components. The suite will be an integration of newly developed components and existing components that are customized and adapted for the Smart Vortex use cases.

The intention of this document is to describe the individual components of the software suite and their relations and interdependencies. Moreover, a software architecture for integrating the different components is proposed, defining integration points and integration mechanisms. The architecture is based on a multi-tiered model, whereby the individual components (and their sub-components) are positioned in different (logical) layers, where the main dependencies between the components are expected to be with the components in the layers immediately above and below respectively.

The architecture has been designed to make it possible to interface external systems, such as PLM/PDM systems and traditional relational database systems. This will be required for future commercial exploitation, where the Smart Vortex suite will have to co-exist with a broad range of existing product lifecycle management systems and various other technical and administrative systems.

The document will be the starting point for detailed technical specifications and implementations of the software components of the Smart Vortex suite.

TABLE OF CONTENTS

1 INTRODUCTION

In the Smart Vortex project, a number of software components will be developed and integrated with existing components to form a suite of software tools known as the *Smart Vortex Suite*. The key software components include a Data Stream Management System (DSMS), data capture and telematics components, visual query and data visualization components, collaboration and decision-making components and social networking components. Data interchange with external systems, such as Product Lifecycle Management (PLM) systems, will also be required.

A key aspect of the architecture is the integration points between the different components. Different mechanisms for integration are chosen depending on which kinds of components are involved, the need for integration and the type of data interchange required. In general, a loose integration between different components is sought, to make it easier to replace components if necessary.

A multi-tier architecture is defined for the Smart Vortex suite. This means that the components (and sub-components) are logically separated into a number of layers, depending on the functionalities and relationships between the components. The reason for using a tiered approach is to facilitate development of flexible and reusable applications, with clear information exchange interfaces. However, the tiered model is more applicable for some components than others (e.g. client/server based application such as web applications).

This document focuses on describing the software architecture of the Smart Vortex suite from a technical (software engineering) perspective. However, there are also other aspects that are important to consider in the design of such a software architecture. Strategically, one other important aspect is the business service dimension, from a process value adding perspective. This aspect of architecture design crosses organizational boundaries. The strategic aspect must consider the fact that large parts of industry will increasingly transition to a service-based business model. These services will often be delivered by a cluster of several cooperating organizations. Therefore, the architecture must be designed so that it is possible to deliver well-defined services based on the software components, i.e. there must be ways of delimiting the functionality into well-defined services that can be delivered independently to the customers.

2 SOFTWARE ARCHITECTURE OVERVIEW

In this section, a high level description of the Smart Vortex suite architecture is given. A multi-tier architecture is defined and the individual components of the suite are positioned within the tiered architecture.

2.1. Software components overview

The Smart Vortex architecture will be composed of a number of software components that are integrated in different ways to form the complete Smart Vortex suite. The software components can be positioned in one of the abovementioned tiers, or they can span multiple tiers. The components can be separated into a number of categories based on the functionality they provide. The following are the main component categories:

- **Data Capture and Telematics components**
  These component captures data from sensors and communicates the data to upper layer components for processing. These are described in section 3.
- **Data Stream Management System**
  The Data Stream Management System is the core component for searching and processing data streams. The DSMS architecture is described in section 4.
- **Visual and Multimodal Query components and Data Presentation components**
  These are the components for formulating queries using multimodal or visual GUI components and components for the presentation of data streams to users. These components are described in section 5.
- **Social Network components**
  These are the software components managing relations between participants of collaborative teams and other applications of the social ontology. Section 6 describes the architecture of the Social Network components.
- **Collaboration and Decision-making components**
  These components include both synchronous and asynchronous tools to support collaboration within a group. The synchronous tools represent real-time interpersonal communication tools, e.g. audio/video conferencing, text chat, tools for managing shared data and collaboration spaces. The asynchronous tools represent shared document repositories and tools for team management and user presence. Other tools include tools supporting decision-making processes, such as brainstorming, voting and analysis tools. The architecture is described in section 7.

A high-level overview of the main architectural components of the Smart Vortex suite is given in Figure 1 below.

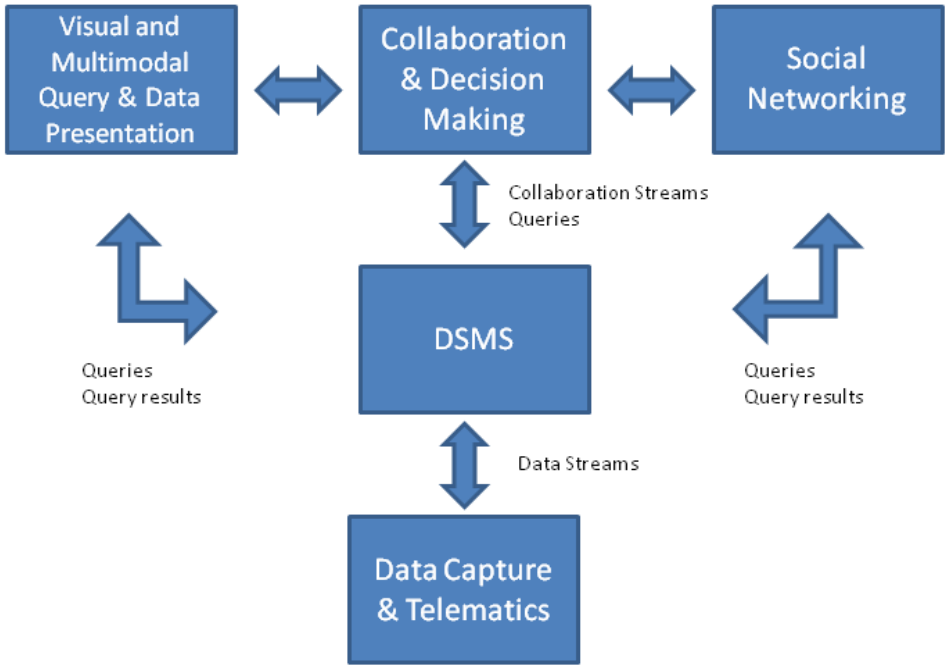

Figure 1. A high-level overview of the relations between the main component groups of the Smart Vortex suite architecture.

2.2. Multi-tiered architecture

The Smart Vortex software architecture is based on a tiered model including the following tiers: Presentation Tier, Application Server Tier, Distributed Augmentation and Annotation Tier, Middleware Tier, Data Integration Tier and Data Tier (see Figure 2).

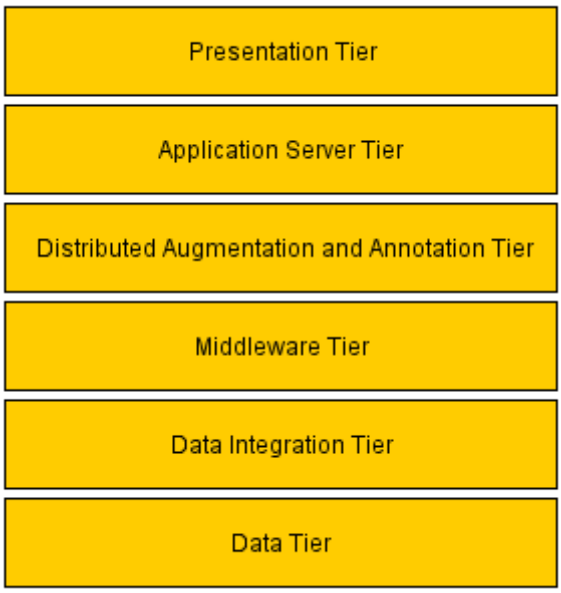

Figure 2: Smart Vortex multi-tier architecture

The reason for the multi-tiered approach is to separate the functionality of different components logically, promoting reusability, modularity and flexibility. Each of the tiers are described below.

2.2.1. Presentation Tier

The presentation tier contains user interface components. This involves presentation of data in different ways including visualization of data. It also contains application control functionality, e.g. traditional graphical user interfaces (GUI) controlled via mouse and keyboard, as well as any other multi-modal user interface.

2.2.2. Application Server Tier

The application server tier contains the logic providing the server side functionality of a client/server application. Specifically, for web-based applications this is where the web-server integration and server-side processing components like servlets or Common Gateway Interface (CGI) components reside.

2.2.3. Distributed Augmentation and Annotation Tier

The functionality of this tier is specific for the Smart Vortex architecture and contains mechanisms for distributing, augmenting and annotating data streams.

2.2.4. Middleware Tier

The middleware tier contains the functionality needed to make multiple components interoperable, by means of well-defined data exchange interfaces. Components of the middleware tier is typically implemented as plug-ins, dynamically linked libraries or other modules with well-defined interfaces, with the primary purpose of providing interoperability and service integration. The middleware tier is where many of the integration points of the Smart Vortex architecture will be located.

2.2.5. Data Integration Tier

The data integration tier supports the combination of data from different sources and provides a unified view of these data sources. Typical mechanisms employed are wrapper interfaces and data hiding techniques. The Data Integration tier provides integration on the data level, rather than at the application component or service level.

2.2.6. Data Tier

The data tier is concerned with data sources, such as storage and retrieval from databases. In the Smart Vortex context, access to data streams is a core functionality of the data tier. Conceptually, the data tier is also where the Smart Vortex information models [1, 2], such as RDF-descriptions of sensors and collaboration data streams, as well as the social ontology resides.

2.3. Smart Vortex components in the Tiered Architecture

In Figure 3, the main software components of the Smart Vortex architecture are presented in relation to the tiered model. Note that this is only a rough depiction of where the main functional components are to be found in relation to the tiers for the purpose of providing an overview. Some of the more prominent sub-components of each component are also shown; these will be described in more detail in sections 3 to 7.

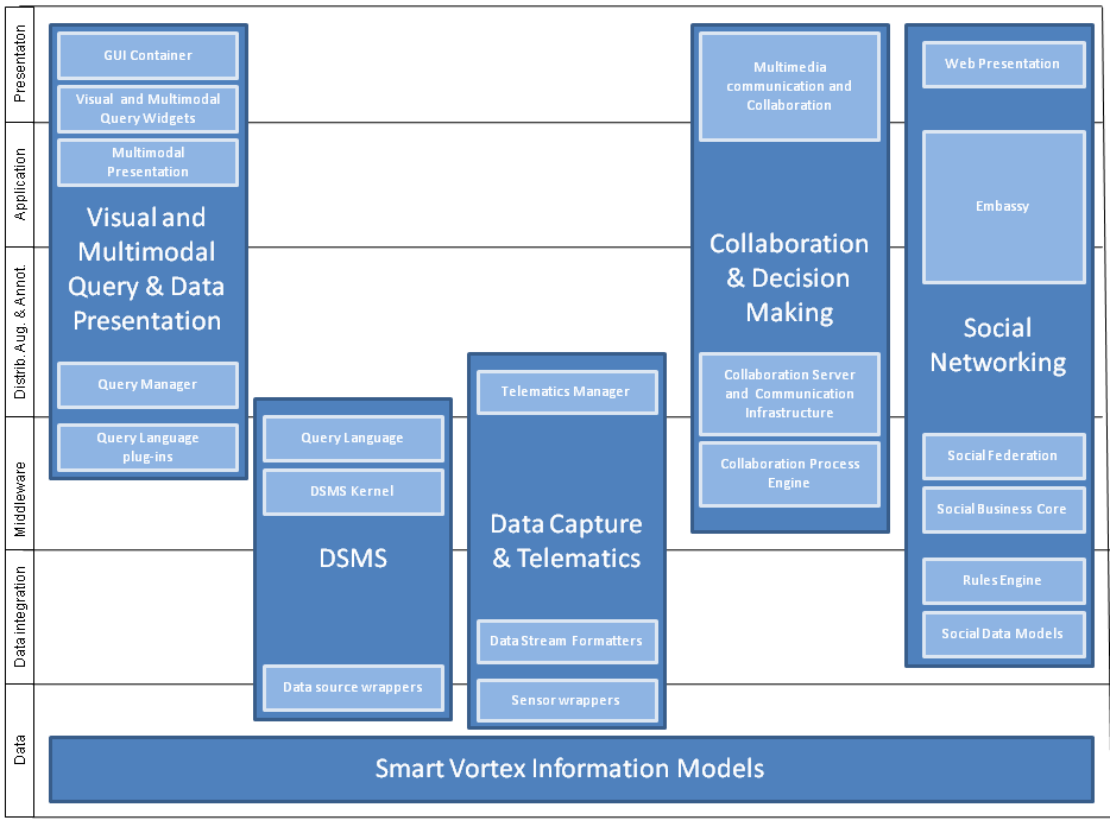

Figure 3. The main functional components of the Smart Vortex architecture, positioned roughly in relation to the tiered model. Some of the sub-components of each component are also shown.

3 DATA CAPTURE AND TELEMATICS COMPONENTS

The Data Capture and Telematics components are the software components that access data from sensors and actuators in equipment and communicate this data to the upper tiers of the architecture for processing (i.e. using the DSMS components) and collaborative analysis or visualization.

The Data Capture components are based on a Sensor Wrapper Interface, which encapsulates the complexities of communicating with the individual sensors, providing an API to the Telematics Module. The Telematics Module manages measurement assignments via the Task Manager and Task Execution Engine and communicates data and measurement assignments with upper tier components of the architecture.

The Data Capture and Telematics components are located at the bottom four tiers (Data, Data Integration, Middleware and Distributed Augmentation and Annotation) of the architecture, as shown in Figure 3 of section 2. The Data Capture and Telematics components can be separated into two modules: the Telematics Module and the Data Capture Module, as shown in Figure 4 below.

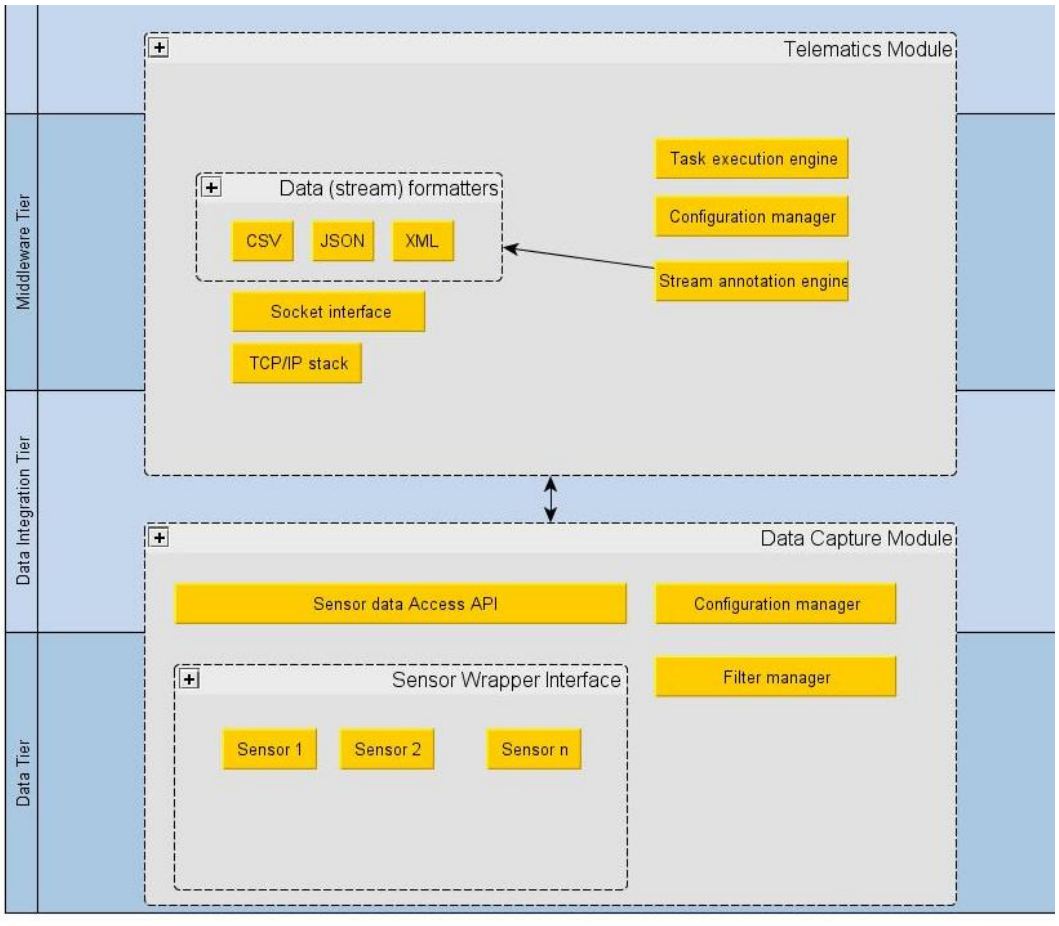

Figure 4. Data Capture and Telematics components.

3.1. The Data Capture Module

The Data Capture module is typically executed in an embedded system in (or near) a machine being monitored. The sensors providing the actual data of the monitoring process are connected directly to this equipment. The configuration of the sensors is done through the Configuration Manager (CM). The CM communicates through the Telematics component

with the upper tiers of the architecture to enable remote configuration of the Data Capture module.

The access to sensor data is provided through the Sensor Data Access API, which is typically accessed by linking the Data Capture Module with the Telematics Module. The Filter Manager (FM) is used for specifying filter masks, deciding which data to deliver to the Telematics Module via the Sensor Data Access API, when data reduction at the source is necessary. The FM is accessed through the Sensor Data Access API, though dedicated function calls (e.g. InstallFilter(), RemoveFilter(), etc).

3.2. The Telematics Module

The Telematics module handles communication of data between the Data Capture module and the upper tier components (e.g. DSMS, Collaboration Server). The Telematics Module can deliver real time data streams, formatted in different ways, depending on the kind of processing that is desired. The formatting is performed by the Data Stream Formatter plug-ins, which expose a TCP socket interface to upper tier components. The format of the data of the streams is decided on a case by case basis. For example, some streams may produce events in JSON format while other streams use CSV.

The raw data streams from sensors can also be augmented with meta-data. This is done by the Stream Annotation Engine. For example, each measurement data sample can be time stamped to record the sampling time of the sensor.

The operation of the Data Capture Process is controlled by the Task Execution Engine (TEE). The TEE manages measurement assignments, designed by the user (engineer), specifying what signal sources are requested, trigger conditions (i.e. when to start/stop measuring) and Data Stream interface set-up parameters. The TEE communicates with the CM for configuration of the signal sources and with the data stream formatters for setting up the data stream interfaces.

4 DATA STREAM MANAGEMENT SYSTEM

The Data Stream Management System (DSMS) will be based on a federated architecture outlined in Figure 5 below. The boxes named *Applications* and *Data sources* are not part of the DSMS architecture itself; these are the main integration points with other components described in this document, and these interfaces are where data flows in and out of the DSMS. The applications are typically visualizers, query formulators and data analyzers that access the DSMS through the External APIs. The data sources can be both streams such as CSV feeds, JSON feeds and access to repositories such as relational DBMSs, files, etc.

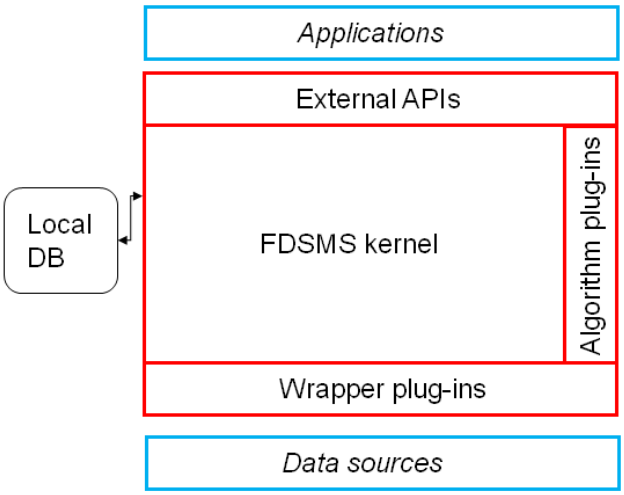

Figure 5. Federated Data Stream Management System (FDSMS) architecture.

The DSMS provides a plug-in mechanism that can be used to implement specific algorithms, like indexing, computations, matching, optimization and classification functions. Plug-ins for Python and Java engines are available, so that algorithms written in these languages can be conveniently accessed by the DSMS kernel.

4.1. DSMS Kernel

The DSMS kernel consists of three main components: the Query Processor, the Distribution Engine and the Query Plan Interpreter, as shown in Figure 6.

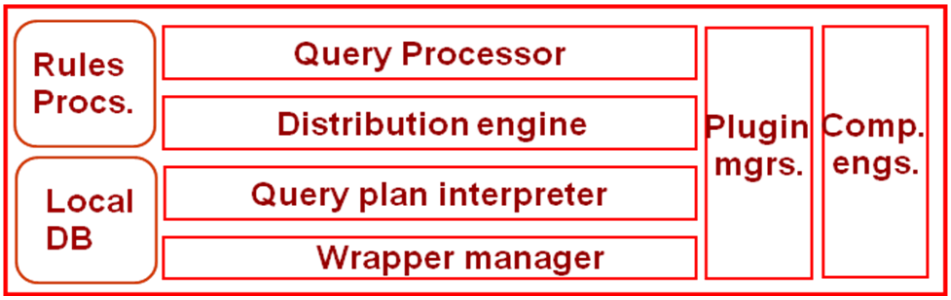

Figure 6. DSMS kernel architecture

The Query Processor receives Continuous Queries specified in a Query Language through the external API. The queries are distributed over a number of processing cores by the Distribution Engine. In each node the query processor translates the continuous queries into an execution plan which is interpreted by the Query Plan Interpreter. The Plugin Managers process plug-in modules that extend the functionality of the DSMS core components. To execute these plug-in modules, different Computational Engines are interfaced, e.g. for Python and Java. For example, an execution plan usually contains calls to foreign functions,

which are plug-ins written in e.g. Python. Execution plans in different instances of the DSMS engine can communicate through system foreign functions. The DSMS engine has its own internal main-memory database where state information about processed streams is stored. Rules and procedures installed in the system execute the continuous queries.

4.2. External APIs

The external APIs are the interfaces the applications are using to access the DSMS. This is done by sending a query formulated in one of the supported query languages. For streaming data this is a Continuous Query. The query languages that will be supported in Smart Vortex will be described in Deliverable D3.2 *Query language survey and selection criteria* [6], and the necessary extensions for formulated continuous queries will be specified in D3.3 *Specification of query language extensions* [7]. Examples of possible query languages to support are SCSQL, CQL, SQL or SPARQL.

4.3. Wrapper interfaces

The wrapper interface encapsulates the plug-ins used by the DSMS to access data sources. Depending on the kind of data source that needs to be accessed, a new data source plug-in can be developed. General data source interfaces based on CSV (comma separated values), JSON and RDB interfaces will be provided.

An example of what the complete DSMS architecture can look like, with different external APIs, data source wrappers, engines and plug-ins is shown in Figure 7 below.

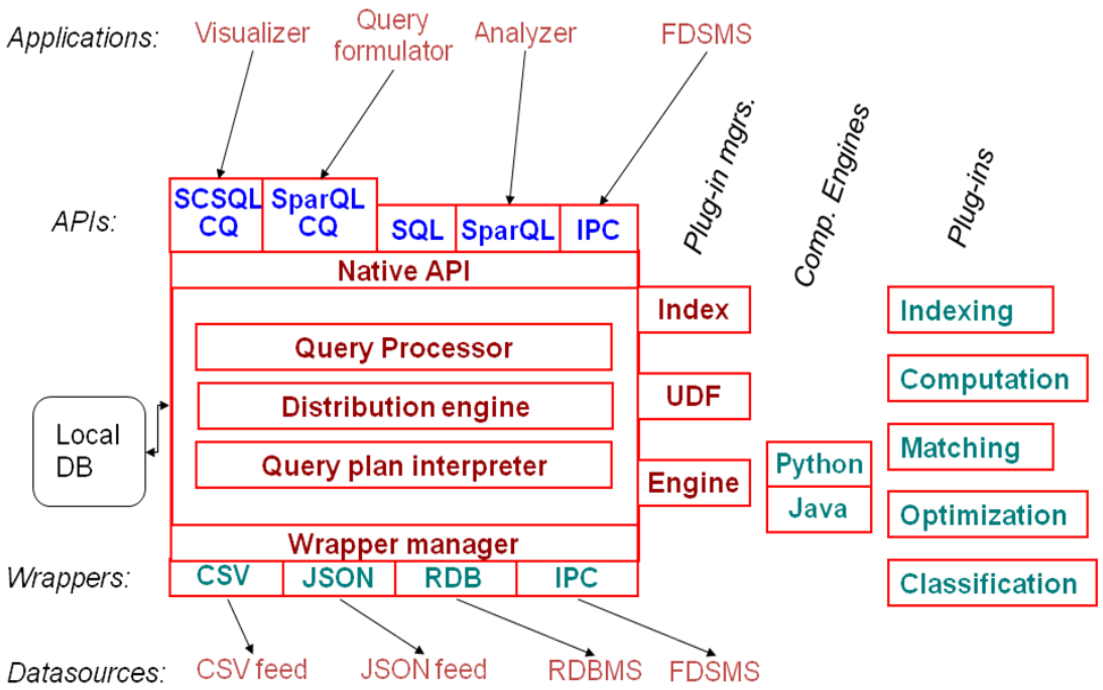

Figure 7. Complete DSMS architecture example

5 VISUAL AND MULTIMODAL QUERY COMPONENTS AND DATA PRESENTATION COMPONENTS

This section presents the architecture of the visual and multimodal query and data presentation components. For the GUI side, the main idea is that the GUI components for visual queries and visualization are composed of many plug-ins that can be plugged in to a general container. It also exposes an interface to capture multimodal input and allows for different presentation modes using other modalities. This widget will provide features to formulate visual and multimodal queries, load pre-defined stored queries, provide access to different presentation modalities and work in a collaborative environment.

The GUI allows users to access the system through icons rather than text commands, exploiting the concept of direct manipulation [8].

In terms of the tiered Smart Vortex architecture, the GUI Container and Visual Query Widgets are located at the Presentation tier, whereas the underlying logic (i.e. the Query Manager) is in the Middleware tier (cf. Figure 3, section 2).

A detailed view of the Presentation tier components is given in Figure 8.

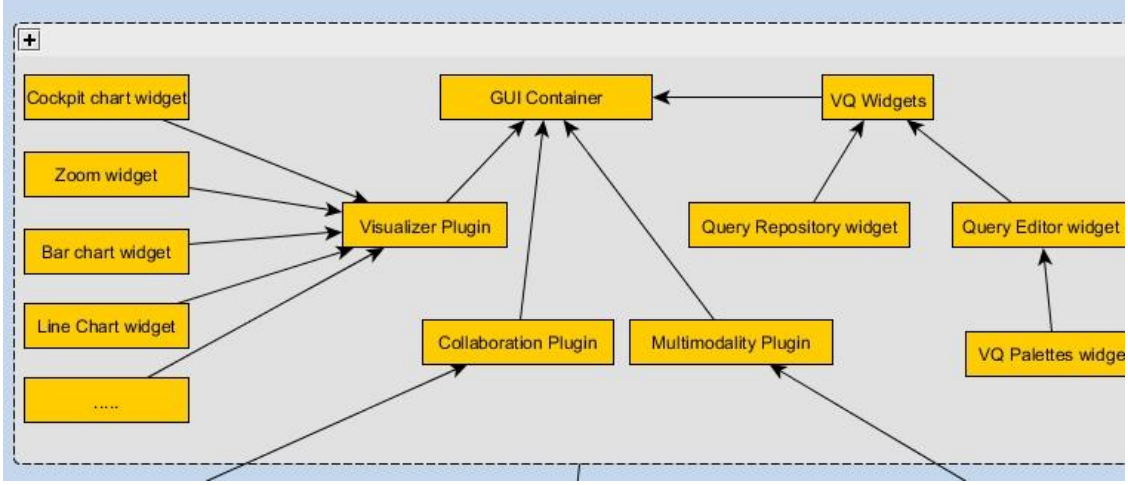

Figure 8: Smart Vortex GUI architecture

In the Middleware tier, the Query Manager is responsible for performing query translations from the graphical representation into the query language syntax which is supported by the DSMS (see section 4.2). The Query Manager has a plug-in framework for supporting different query languages. This is illustrated in Figure 9.

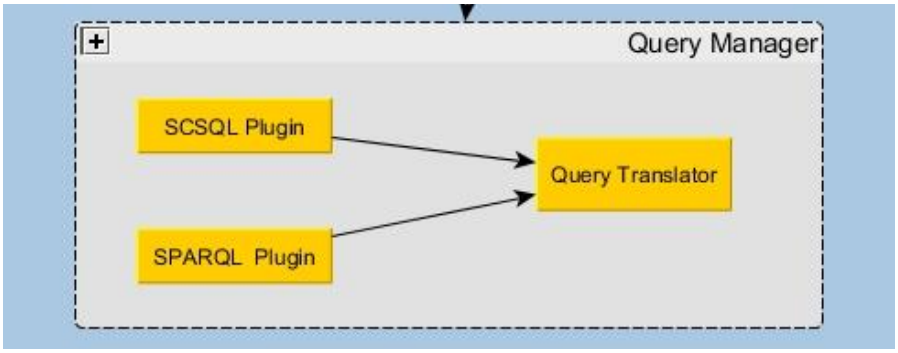

Figure 9: Smart Vortex Query Manager architecture

Each of the Visual Query and Data Visualization components are described below.

5.1. GUI Container

The GUI Container is the core of the graphical user interface. Different plug-ins can be selected to visualize data streams, write and run visual queries or work in a collaborative space.

5.2. Visualizer Plug-in

This module provides graphical icons to perform visual analytics tasks. Several features will be implemented, such as to visualize the whole data set, filter the part of interest, and zoom on the important data to have a deep understanding of the current situation.

5.3. Cockpit Chart widget

The cockpit chart widget visualizes multiple data stream sources in a compact mode. This widget provides an overview of the system, visualizing many sensors simultaneously.

5.4. Zoom Widget

The Zoom Widget provides a zoom on a specific sensor/machine through graphical indicators (colors, shapes, etc.).

5.5. Visual Query Plug-in

This module provides graphical tools to build visual query. Analysts can formulate queries through icons manipulation; each icon has a translation in one or more query languages.

5.6. Query Repository Widget

This widget provides access to stored queries. Engineers can write and store the queries they expect are useful to analysts for a standard working process. Analysts can also write specific queries when needed through the component Query Editor Plug-in (see below) and then save the customized query permanently in the system.

5.7. Query Editor Plug-in

This widget allows analysts to define new visual queries. There are two main ways to formulate a new query: modifying an existing Stored Query or starting from an empty form. In order to create a new query this module uses the features provided by the Visual Query Palettes.

5.8. Visual Query Palettes Widget

The Visual Query Palettes Widget contains the graphical icons needed to build the visual query. The palettes contain a wide range of operators.

5.9. Collaboration Plug-in

This component communicates with the collaboration infrastructure (see section 7). It manages the data sharing and the analysis interaction (e.g. audio, video, brainstorming and voting).

5.10. Multimodality Plug-in

This component exposes an interface to capture input coming from different modalities such as multitouch, voice, etc, to work with the queries but allow also to present the results of query execution. For instance, spatial audio can be used to deliver more information to the users.

5.11. Query Manager

The Query Manager is responsible for the query translation from the graphical representation to the query language syntax and for the query execution.

5.11.1. Query Translator

The Query Translator module is responsible for the translation process from the graphical query to a specific query language. It is an extensible module, so languages can be added by building new plug-ins. The set of query languages to support will be decided based on the recommendations in Deliverable D3.2 *Query language survey and selection criteria* [6].

For each supported query language a query language plug-in will have to be developed. Two main candidates for such query language plug-ins are a SCSQL plug-in and a SPARQL plug-in (as shown in Figure 9).

5.11.2. SCSQL Plug-in

This plug-in translates the visual query to the SCSQL language.

5.11.3. SPARQL Plug-in

This plug-in translates the visual query to the SPARQL language.

6 SOCIAL NETWORK COMPONENTS

This section presents the architecture of the social network components provided by FTK. The principle behind these components is the idea that collaborative environments form social networks and that it is therefore possible to raise the collaboration support by providing paradigms from online social networks (OSNs).

Figure 10 provides a rough overview of these parts.

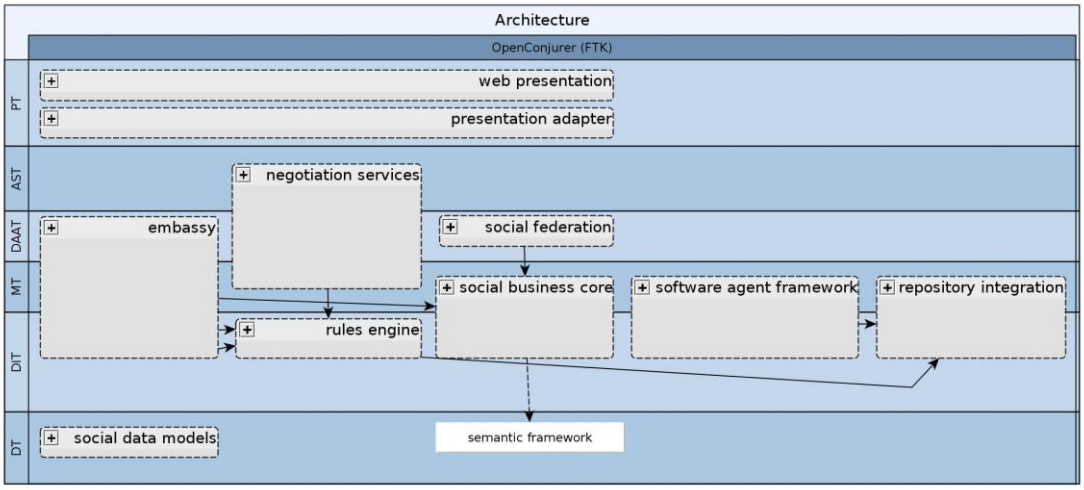

Figure 10: The social network components

The box labelled “semantic framework” in Figure 10 represents the application of current third-party components for persistising and reasoning using OWL technology. Typical frameworks are Jena, Protegé and Sesame. Currently these are being thoroughly evaluated to identify the optimal solution for Smart Vortex. Because of the Open Source strategy of the project, the free availability and licensing will be taken into consideration as well as the functionality. All the remaining boxes will be contributions from FTK which, in their entirety, are often referred to as “OpenConjurer.”

Aside from providing services for social information, this part of the Smart Vortex suite will also offer services for the cross-organisational execution of tasks including the automatic mapping of data models.

Regarding multi-tiered architectures, OpenConjurer occupies all six tiers defined earlier in this document, namely the Data Tier (DT), Data Integration Tier (DIT), Middleware Tier (MT), Distributed Augmentation and Annotation Tier (DAAT), Application Server Tier (AST) and Presentation Tier (PT).

The following subsections describe the tasks of the individual components. All components will be fully specified in D8.1 and their implementations will be part of D8.2.

6.1. Social Data Models

There are different ways to describe the relationship between entities in general and people in particular. These have to be represented to enable the annotation done in other components. As they belong technically to the description of data sources rather than the integration of data itself, they belong to the Data Tier.

Most importantly, this component contains a *generic social data model* that is used to describe domain-specific models in a uniform manner.

6.2. Embassy

The Embassy is the central mechanism that allows the cross-organisational execution of tasks while upholding (mutual) security policies within the participating organisations. It also ensures that data is properly translated between the different data models.

This algorithm was developed and described in [13] and uses the Rules Engine described below.

The component provides authorization and authentication services as well as mapping and delegation services for the propagation of activities to other instances.

6.3. Social Business Core

The Social Business Core (SBC) offers an interface for querying and manipulating social context data. It is used by other components as a basic storage. The mechanism of the core uses semantic web technology for reasoning over ontological data.

The SBC mainly consists of a *social directory service* and an *ontology engine wrapper*. The purpose of the latter is to allow for different semantic frameworks to be usable.

6.4. Social Federation

The Social Federation component allows to combine or to map two social ontologies, e.g. of two different organisations. It is used to set up federation rules and to annotate processes in such a way that remote servers can act on the social context appropriately.

The module consists of a set of protocols for the establishment of relationships between organisations, as well as a *social annotation service* responsible for capturing socially contextualised provenance information.

6.5. Repository Integration

Repositories are more than mere data storages. They often allow for very complex queries. These powerful repository mechanisms are essential for larger enterprises. The Repository Integration module allows for a seamless integration of repositories into the Smart Vortex suite.

The component hosts two services, one being the so-called *repository service* which can be queried for repository data and the other being a *repository interface* which allows for the utilisation of different repository systems..

6.6. Software Agent Framework

OpenConjurer is not only concerned with human agents, but also with software agents. Agents as used in [13] are entities that can act upon and manipulate an environment. A software agent may be another instance of OpenConjurer, an application using OpenConjurer or a gadget within the website. The Software Agent Framework is used to be able to describe and act on these software agents in a uniform manner.

The framework provides an *agent service* that can be polled for information on different agents, a *software agent model* that contains important metadata on applications and software and a registry to persistise the information.

6.7. Rules Engine

The Embassy hosts several subsystems that act on policies or rules, for example the Authorisation Service. While security policies can be expressed using OWL, it may be necessary in the future to support one or more of the established security languages like REL. For this reason the Rules Engine component is decoupled from the other systems to provide a simple way for exchanging the core security language.

The engine consists of a *policy factory* generating policy objects for the authorisation service, an *inference service* that allows for different policies or conditions to be checked and a *rules persistence interface* that is used for persistising the required data.

6.8. Negotiation services

To make sure that the mapping between data models is done automatically, mappings need to be established. Also the effect for the security policies has to be evaluated. This process is called negotiation and the Negotiation Services component provides the necessary business logic to establish these relations.

The negotiation services consist of a simple negotiation service and an application that houses the business logic needed to map different concepts to each other.

6.9. Presentation Adapter

To allow for different types of GUI (e.g. web, native applications) there needs to exist a uniform way in which software agents may interact with the user in the different environments. Moreover, some software agents may only work in certain environments. The Presentation Adapter provides a way to resolve these conflicts.

The adapter contains mappers for the different actions and protocols used.

6.10. Web Presentation

As a first example of a GUI for the Smart Vortex suite, the Web Presentation component will provide a website as a frontend.

The subcomponents envisioned at this stage are profile and search options as well as the integration of certain collaboration tools.

7 COLLABORATION AND DECISION MAKING COMPONENTS

The collaboration and decision-making components are intended to enable the users of the Smart Vortex suite to perform inter-organizational collaborative work regarding data monitoring, data analysis and visualization. The collaboration processes that will be supported by the system components are described in D2.1, section 3.3.1 [1].

In order to support these collaborative work processes, a number of synchronous and asynchronous collaboration tools will be developed and integrated with existing tools that will be adapted for the specific Smart Vortex use cases.

An overview of the software components for collaboration and decision making, and their positions within the tiered Smart Vortex architecture is shown in Figure 11.

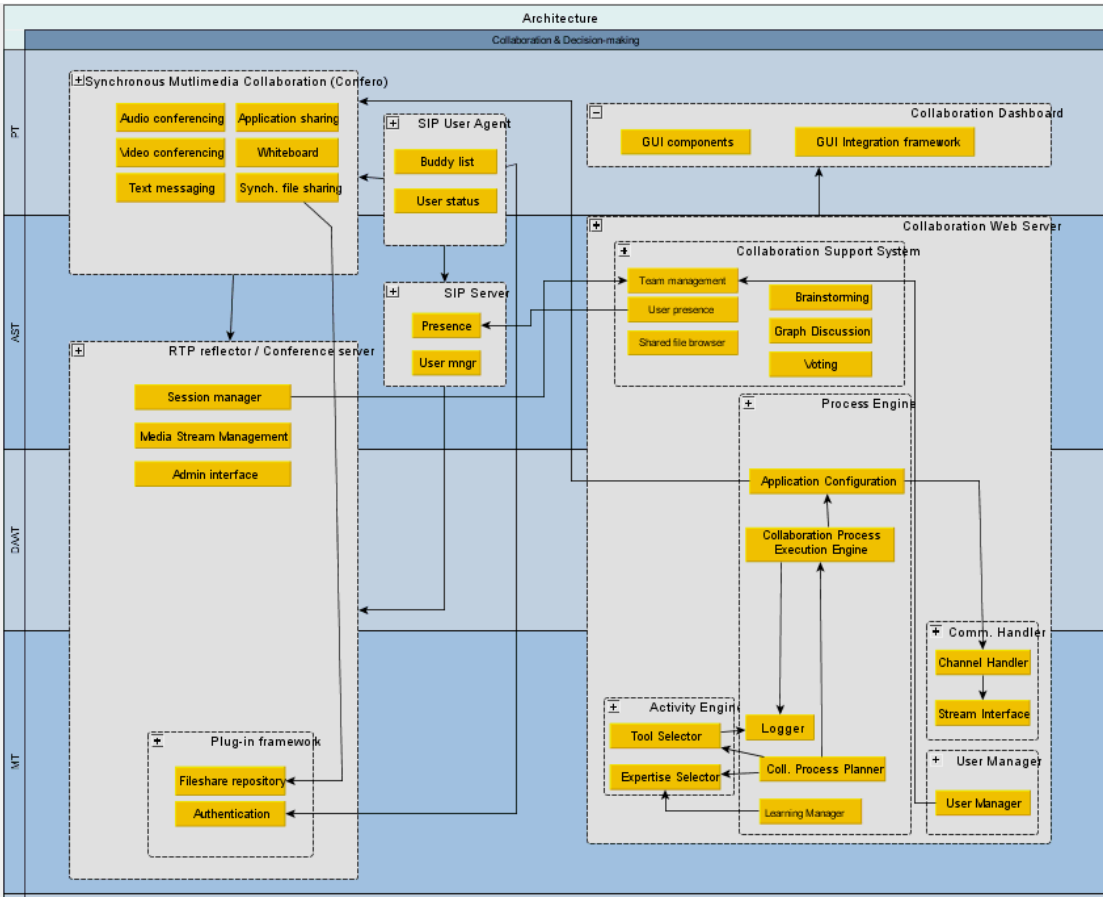

Figure 11. Collaboration and Decision-making components

7.1. Synchronous collaboration support

The synchronous collaboration tools will support real-time audio and video communication, general application sharing, file sharing, instant text messaging, shared whiteboard, session initiation and session management. The software collaboration tool Alkit Confero [4], developed by Alkit Communications, will be adapted and customized to fit in the Smart Vortex suite architecture. Integration of other synchronous collaboration tools (e.g. Viber, Skype) will also be possible.

7.1.1. Synchronous multimedia communication

The synchronous multimedia communication will rely on a client application, which is at the Presentation Tier of the Smart Vortex architecture (top left in Figure 11). A customization of Alkit Confero is intended to provide this functionality.

Session initiation and management will be implemented using the Session Initiation Protocol (SIP) [5]. This will require SIP User Agents and SIP servers, where users are registered. This user management should be integrated with the team management functions intended for the Collaboration Server / Dashboard component (see below). Alkit Confero provides a SIP User Agent that can be used to implement the presentation layer services of the session initiation and control.

Furthermore, the synchronous collaboration support will require a conference server component for relaying media streams between the participants in real time. An RTP reflector will be used for this purpose. Synchronous conferencing sessions will be set-up on the reflector using the SIP protocol. The reflector will have a plug-in framework, providing a good integration point with other components of the Smart Vortex architecture, e.g. for user management, authorization mechanisms, process support, etc. Media streams can also be communicated to the DSMS for processing by means of a customized reflector plug-in.

7.2. Asynchronous collaboration support

The asynchronous collaboration support of the Smart Vortex architecture is developed using Web-based components that give the user access to collaborative data and team management functionality. These functions are to be implemented in a Collaboration Server, where collaboration data is accessible between synchronous collaboration sessions (i.e. virtual meetings). The Collaboration Server should also provide mechanisms for transitioning between asynchronous and synchronous modes of collaborative work, e.g. by inviting team members to meetings, launching synchronous collaboration tools, etc.

The collaboration team set-up and management functions should be supported by the Social Networking components, so that the social graph of a collaborative project can be searched for competence and skills matching a collaboration process.

7.3. The Dashboard

The Dashboard is intended to be a web-based user interface component providing the presentation side of the Collaboration Server, wherein the users collaborate more or less in real time and have real-time or near real-time access to monitoring data and shared visualization of data. The Dashboard provides an integration point for the Visual Query components (see section 5.1), for posing continuous queries to the DSMS and visualizing the results collaboratively. The Dashboard also provides an integration point between the asynchronous collaboration tools and the synchronous collaboration tools. The synchronous multimedia collaboration tools (e.g. Alkit Confero), should be possible to launch from the Dashboard. The Dashboard, together with the synchronous multimedia collaboration tools can be seen as a "Virtual Collaboration Room" in the Presentation Tier of the Smart Vortex architecture. A mock-up view of what this could look like is shown in Figure 12.

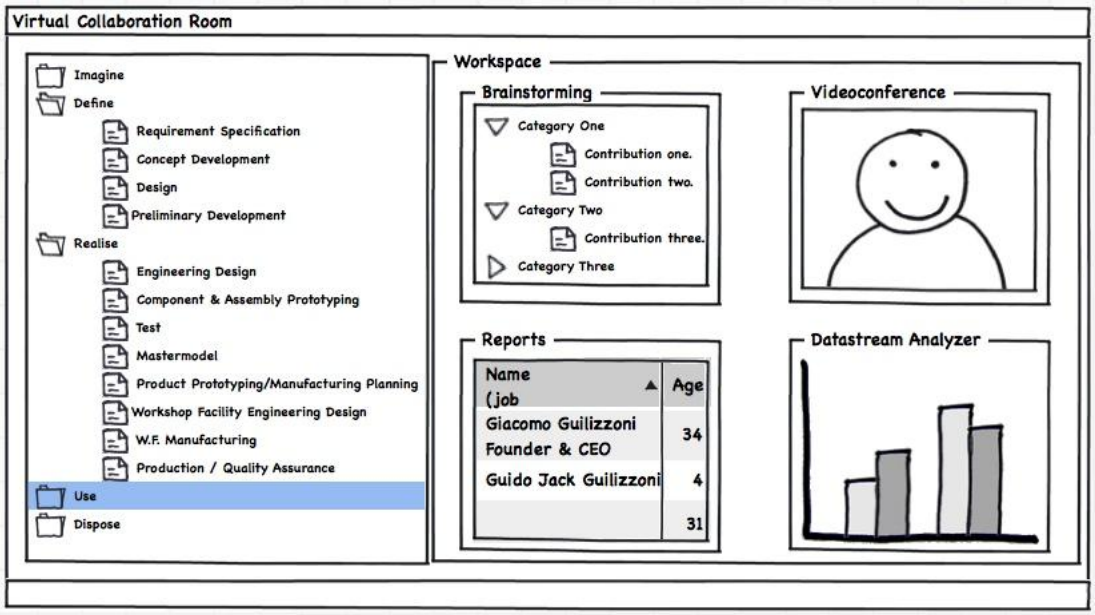

Figure 12. Virtual Collaboration Room or "Dashboard". This mock-up illustrates how the presentation tier components for Collaboration & Decision-making can be loosely integrated with the Visual Query & Data Visualization components to form a unified view of the system for an end user. Similar user interface integration can be done with the Social Network components, for instance for visualizing relations within a collaborative team.

7.4. Collaboration Process Support

A collaboration process model represents the sequence of activities that a group performs towards a goal. Such models are important to describe the way people interact with the systems and with each other to accomplish a certain goal.

Facilitators generally guide a group through collaboration processes. They are experts that have knowledge on conducting a collaboration session and guiding groups in the use of collaboration support systems. The facilitator selects the most appropriate tools for the group during a collaboration session and provides them with instructions for its correct use. However, facilitators might not always be available to support a group in a task, especially in crisis response situations, such as a sudden need for equipment maintenance.

In section 4.2.5 of Deliverable D1.1 [9] is described a mechanism that allows a company (Volvo CE in this case) to perform remote diagnosis of malfunctioning machines. Such a mechanism allows the company to save travel costs by not sending experts to the location of the equipment. Besides such gain, the remote infrastructure allows different experts to participate in a diagnosis process in difficult cases. From the collaboration perspective, this new method also allows the group to profit from the benefits of following collaboration processes, to achieve more effectively the goals of the group. However, such sudden diagnosis situations have a dynamic characteristic and are difficult to predict. Therefore it is not possible to plan in anticipation the attendance of a facilitator to guide experts during a collaborative maintenance process.

One possible approach to overcome such a limitation is to capture the experience of professional facilitators within an intelligent collaboration support system so that this can act as a professional facilitator [10, 11, 12] and empower persons to effectively collaborate with each other. The main idea is that such a system (1) analyses the context within a given collaboration by using information provided from data streams, (2) selects appropriate experts

to tackle a current problem, (3) recommends an appropriate process and tool support and (4) observes the collaboration process to ensure effective collaboration by adapting the process and tool support according to changes in the context.

The architecture of such a collaboration process support system is shown in Figure 13.

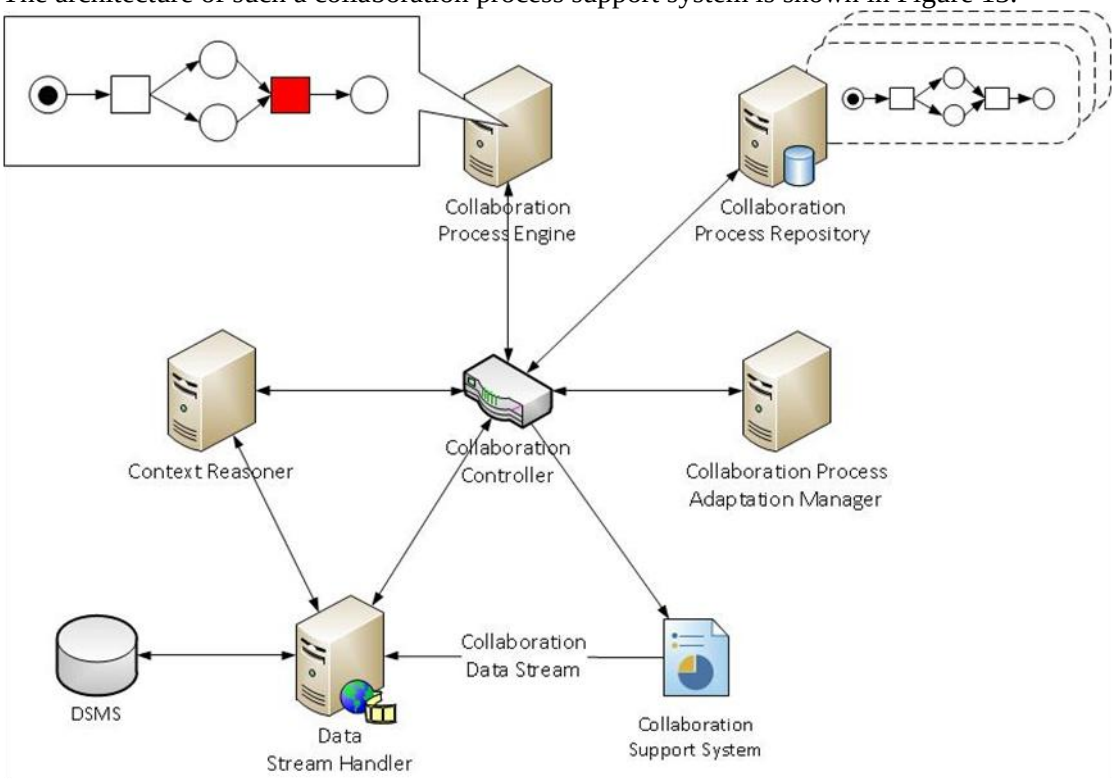

Figure 13: Architecture for execution of collaboration processes.

The architecture of this system basically contains entities to store and execute collaboration process models, to manage context information resulting from data streams that is relevant for the collaborative session, to interface with a DSMS gathering relevant equipment data for the collaboration session and to configure suitable tools that are relevant for the experts in a group.

The implementation of the collaboration process support mechanisms is performed in the Collaboration Server, with the GUI components integrated in the Collaboration Dashboard (see right hand side of Figure 10).

The following subsections describe each entity in detail to clarify their roles towards a collaboration support system.

7.4.1. Collaboration Process Repository

The collaboration process repository (CPR) contains all the collaboration processes that are either designed by a facilitator or were adapted during the course of collaboration. Each collaboration process describes the people involved in the process, the data that should be involved in the process, the contextual information in which the process was used, the tools used during each activity and the variations of the collaboration process that are adapted to suit a specific context.

7.4.2. Context Reasoner

The context reasoner represents the entity that reasons over basic data, transforming it into processed information that we name "context." Context contains analogous definitions such as

the semantic models of data and data streams that are used in the DSMS (see section 2.1, Deliverable D3.1 [3]).

Context represents a similar concept to the DerivedStream. Both context and DerivedStream represent reasoned information over basic data. According to the definition of DerivedStream, basic data is considered machine operational content, such as power consumption (raw data) and equipment location (meta data). However, according to the definition of context, basic data represents data provided by the DSMS, such as equipment data and resulting data from the interaction among experts and the system (collaboration data). Additionally, basic data also represents data concerning the collaboration process engine, such as, executed steps during the collaboration process and planed steps to be executed in the future.

7.4.3. Collaboration Process Engine

The collaboration process engine (CPE) is responsible for reading, interpreting and executing the collaboration process. This entity continuously assesses the course of collaboration and recognizes whenever it has to perform the transition between the activities of a collaboration process model. This entity is also cooperates with a context provider entity, providing the history of actions and activities performed along the collaboration process.

7.4.4. Data Stream Handler

The sensor data stream handler represents an interface to communicate with the DSMS. This entity is responsible for preparing queries for the DSMS. It uses the DSMS query language for it and retrieves data required in the collaboration support system. This entity also gathers data from the collaboration support system and sends it to the DSMS as part of the collaboration data stream.

7.4.5. Collaboration Controller

The collaboration session manager acts as a controller for the collaboration system. It is basically responsible for mediating the communication with the other entities.

7.4.6. Collaboration Process Adaptation Manager

The collaboration process adaptation manager (CPAM) is responsible for selecting collaboration process models and adapting them when necessary. This entity selects the collaboration process models according to the amount of time that a group has to find a solution to prevent a machine breakdown (response time). Whenever the CPAM is aware of the time span a group has to accomplish a task, it selects a collaboration process model. Such selection, by means of rules, is based on a matchmaking process that considers the available response time and the expected time that a collaboration process model is accomplished, in average. The collaboration processes models are selected when their expected time to be accomplished are less than the response time. In this case, the CPAM selects a model from the CPR and sends it to the CPE for its execution.

This entity adapts the flow of activities that are performed by a process and the tools used by each activity, according to a given context situation. For example, a machine is pre-diagnosed as having 1 hour as a remaining useful lifetime (RUL). The maintenance team reacts to such an event following a certain collaboration process model that contains 3 activities, to diagnose its problems, that consumes 45 minutes. However, at a certain point in time, the machine starts to degrade faster than expected and the machine has 40 minutes RUL, rather than the expected 1-hour RUL. To adapt to this situation, the CPAM acts, changing the collaboration process model to suit such a new time span. Among different types of adaptation, one possible is turning 3 activities into 2, to save time.

8. INTERFACING EXTERNAL COMPONENTS

Whereas this document so far mainly has been concerned with how to develop and integrate a number of components being part of the Smart Vortex suite, another important question is how to interface external systems and to import/export data from/to them.

A number of highly relevant systems, external to the Smart Vortex suite, which will need to be interfaced are the following:

- Sensor equipment and data capture devices
- PLM/PDM systems
- Relational Database Management Systems (RDBMS)

8.1. Sensor equipment and data capture devices

These are devices that are needed to access data from sensors or actuators in machines being monitored. Examples include CAN interfaces and various sensors. These will be interfaced from the Smart Vortex Data Capture module described is section 3.1. The approach for hiding the details of each specific device is to write a wrapper interface providing one generic interface for all kinds of capture devices.

8.2. PLM and PDM systems

Data from existing Product Lifecycle Management systems (PLM) and Product Data Management systems (PDM) will typically need to be accessed, processed, collaborated upon, updated and stored. This can be done in a multitude of ways depending on which kind of data it is and what data exchange interfaces are supported by the specific PLM/PDM system. The Smart Vortex architecture will employ a pragmatic approach for data exchange with PLM/PDM systems. The focus will not be on the data exchange mechanisms *per se*, but rather on the information content that needs to be communicated and the principles involved.

In general, data can be accessed from PLM/PDM systems by means of traditional Relational DBMS access mechanisms (see below). The DSMS, being the core data management component of Smart Vortex, supports access to RDBMS (see section 4.3).

8.3. Relational Database Management Systems

Not only databases for PLM/PDM systems, but also other kinds of database systems (e.g. administrative, financial, etc) may need to be accessed by the Smart Vortex suite components. This will be handled by traditional RDBMS access mechanism, e.g. based on ODBC and SQL.

9. SUMMARY AND CONCLUSIONS

This document has described the software architecture of the Smart Vortex suite. A multi-tiered software architecture approach was chosen in order to make it easy to position each component in relation to the others, and define interfaces between them.

Each of the components of the suite has been described and the main integration points between the components have been identified and integration mechanisms have been suggested. The integration is performed in different ways depending on the type of components concerned and the type of data interchange required.

In the case studies that will be conducted in the project, the Smart Vortex suite components will have to interface external systems and devices. The Smart Vortex architecture has been designed to be able to import and export data to external systems, such as PLM/PDM systems, using established data interchange mechanisms.

REFERENCES

[1] L. Johansson et al. "SMART VORTEX Information model," Smart Vortex Deliverable D2.1, June 2011.

[2] T. Risch et al. "An RDF-based model describing streams and other data from products, social networking systems, collaboration and products in use for scenarios," Smart Vortex Deliverable D2.2, June 2011.

[3] T. Risch et al. "Semantic models of data streams," Smart Vortex Deliverable D3.1, June 2011.

[4] M. Johanson, "Multimedia Communication, Collaboration and Conferencing using Alkit Confero," Alkit technical report 2004:1.

[5] J. Rosenberg et al. "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002.

[6] T. Risch et al. "Query language survey and selection criteria," Smart Vortex Deliverable D3.2, September 2011.

[7] T. Risch et al. "Specification of query language extensions," Smart Vortex Deliverable D3.3, March 2012.

[8] B. Shneiderman, "Direct Manipulation. A Step Beyond Programming Languages," IEEE Transactions on Computers, Vol. 16, No. 8, August 1983, pp. 57–69.

[9] H.-U. Heidbrink et al. "Survey of users in Europe and Smart Vortex usage scenarios and cooperation support," Smart Vortex Deliverable D1.1, June 2011.

[10] R. O. Briggs, et al. "Facilitator in a Box: Computer Assisted Collaboration Engineering and Process Support Systems for Rapid Development of Collaborative Applications for High-Value Tasks," Hawaii International Conference on System Sciences (HICSS-43), IEEE computer society, 2010.

[11] G. Kolfschoten, and S. Lukosch, "Intelligent Collaboration Support: a conceptual framework for a new generation of collaboration support systems," Group Decision and Negotiation 2010, 2010, pp. 38-40.

[12] S. Lukosch, and G. Kolfschoten, "Towards effective collaborative design and engineering," Proceedings of the 2nd Workshop on Semantic Models for Adaptive Interactive Systems, 2011.

[13] J. Grabarske, “Use cases and security requirements for data exchange between corporate social networks,” master thesis, Fernuniversität Hagen, April 2011.