FInest – **F**uture **In**ternet **e**nabled optimi**s**ation of **t**ransport and logistics networks

Deliverable D3.3

Interim Technical Specification for the Domain-Specific Future Internet (FI) Platform for Transport and Logistics

Project Acronym	FInest	
Project Title	Future Internet enabled optimisation of transport and logistics networks	
Project Number	285598	
Workpackage	WP3 (Solution Design and Technical Architecture)	
Lead Beneficiary	SAP	
Editor	Michael Stollberg	SAP
Contributors	René Fleischhauer	SAP
	Michael Stollberg	SAP
	Marianne Hagaseth	MRTK
	Andreas Metzger	UDE
	Clarissa Marquezan	UDE
	Guy Sharon	IBM

	Fabiana Fourier	IBM
	Özgür Sönmezer	KOC
	Gökhan İşleyen	KOC
Reviewers	Rod Franklin	KN
	Oyvind Olsen	NCL
	Kay Fjørtoft	MRTK
Dissemination Level	PU	
Contractual Delivery Date	30.09.2012	
Actual Delivery Date	30.09.2012	
Version	1.0	

Abstract

This report presents the third Deliverable of work package WP3 "Solution Design and Technical Architecture" that is concerned with the overall design and architectural specification of the collaboration & integration platform for transport and logistics business networks (the FInest Platform). In continuation of the preceding report deliverable, it presents the refinement of the overall vision of the FInest Platform, reports on the activities for cross-WP alignment and coordination that have been undertaken in the period M7-M12 of the project, and presents the progress towards the overall technical architecture. With this, the Deliverable presents the work progress on Tasks T3.1 – T3.4 as defined in the DoW of the project.

Based upon and in continuation of the results as presented in the preceding reports of WP3, the work on the overall solution design and technical architecture of the FInest Platform for the M18 milestone of the project has focused on (a) the elaboration of the interim technical specification of the FInest Platform, therewith performing the major step from the conceptual design presented at M12 towards technical specifications as the basis for implementation, and (b) on the continued refinement of the overall concept, in particular with refining the overall operational model of the FInest Platform in order to properly address the requirements and expectations identified in WP1 and WP2 and the extension of the overall concept with additional technical building blocks that appear to be needed in order to allow for the proper operation and usage of the FInest Platform, and prepare for its commercialization. The work has been conducted in two parallel work streams, with continuous and iterative feedback and assessment by the domain experts involved in the project.

This report presents the results of both work streams in a condense manner, including:

- *The refined functional specification of the FInest Platform as a whole, explaining functionalities of the main building blocks and how they can be used and combined in order to enable efficient and seamless collaboration in transport & logistics networks;*
- *The identification and initial technical design considerations of additional technical building blocks, in particular: a Store for the services and apps provided in the FInest Platform as the element for ensuring commercialization, Component Management and Middleware technologies to allow for a proper functional operation of the FInest Platform on the Cloud, and the refined concepts for enabling the personalization and customization of the FInest Platform for individual business needs;*
- *The interim technical specification of the FInest Platform with respect to the interaction of the four Core Modules designed in the project as the unifying and integrating technical architecture across work packages WP5-WP8, along with the methodology and roadmap that is applied for the technical design work stream;*
- *The refined design and interim technical specification of the technical building blocks of the FInest Platform that have already been identified previously, namely the Front-End as the main point of access for End-Users offering an 'all you need in one place' user experience, the Back-End layer that supports the integration of existing and external systems, and the Security Framework that ensures secure and reliable handling of business information on the FInest Platform.*

Document History

Version	Date	Comments
V0.1	06-08-2012	TOC + Content Outline
V0.2	10-08-2012	Content Outline Sec 3
V0.3	16-08-2012	Revised Structure Sec 4
V0.4	26-08-2012	Integrated Inputs (Sec 4)
V0.5	12-09-2012	Restructured deliverable
V0.6	13-09-2012	Added Partner Inputs (Sec 3)
V0.7	18-09-2012	Added Introduction + Content Section 2
V0.8	20-09-2012	Integrated revised content (all sections)
V0.9	21-09-2012	Added Abstract, prepared for internal review
V1.0	28-09-2012	Incorporated review comments, finalized for submission

Table of Contents

List of Tables

List of Figures

Acronyms

Acronym	Explanation
App	Short form for an application running in mobile devices
ASP	Application Service Provisioning (traditional model of software delivery)
B2B	Business to Business
Back-End	Layer of the FInest Platform for facilitating the integration of external systems (legacy & standard business systems, 3rd-party services, ..)
BCM	Business Collaboration Module
BPEL	Business Process Execution Language (OASIS standard)
Cloud	Infrastructure for providing computational resources (servers, virtual machines, etc.) in a centrally management environment; platforms, services, and applications can be deployed on this in order to minimize the TCO (total cost of ownership) as well as other issues relevant for enabling cheap and quick development of high-quality solutions
Collaboration Objects (CO)	Conceptual element for storing transport- and logistics related information within the BCM module
Core Modules	The central functional components of the FInest Platform developed in WP5 – WP8 (BCM, EPM, TPM, ECM)
Demonstrator	An early prototype showcasing how the FInest Platform shall support future (To Be) business scenarios in the transport and logistics domain
DOM	'Data Object Model' (technical term): the technical model of a data object describing its basic structure and data types
ECM	E-Contracting Module
Ecosystem	An economic system comprised of various stakeholders that conduct business together, forming an overall value added network
EPM	Event Processing Module
ERP	'Enterprise Resource Planning' system: overall term for standard business systems for managing the resources of an enterprise (e.g. customers, employees, sales & orders, etc.); often used as synonym for the older solutions from SAP (e.g. SAP R3 sytems, SAP Business Suite)
ETA	Estimated Time of Arrival
FInest Platform	The overall technical solution that is designed in the project
FI-WARE	FI PPP project that develops the 'Future Internet Core Platform', which consists of so-called 'Generic Enablers' that shall be used for realizing the FInest Platform; see public website: www.fi-ware.eu
Front-End	The GUI and access points where end-user interact with the FInest Platform
GDL	'Gadget Description Language': technical annotation language for gadgets (self-contained technical elements that can be added & configured to web-based UIs with respect to individual needs)
GE	Abbreviation for 'Generic Enabler'; term used in the context of the FI PPP, referring to one of the specific generic technologies that are developed in the

	course of the FIWARE project
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technologies
IDM	Identity Management - general term for authentication, authorization ,roles, and privileges/permissions within or across system
Interface	Technical element for (automated) information exchange between components
IoS	'Internet of Services': any kind of service (technical, business, non-technical) that is accessible over the Web and can be re-used in various application scenarios, fostering the idea of service-orientation
IoT	'Internet of Things': technologies for receiving information about real-world objects (e.g. via sensor networks) and enable their treatment as programmatically accessible entities in software environments
IPS	Intrusion Prevention System (system for recognition / prevention of attacks on computer systems)
LDAP	Lightweight Directory Access Protocol, protocol for accessing and maintaining information in distributed directories
OMG	Object Management Group, technology standardization body
PaaS	Platform as a Service
PCS	Port Community System
PKI	Public Key Infrastructure (security technology)
REST Service	A Web Service accessible as a Web Resource, applying to the principles of Representational state transfer (REST)
RFID	"Radio-frequency identification": a IoT technology that uses radio waves to transfer data from an electronic tag (RDIF tag) attached to a real-world object into a software environment via specialized readers
SaaS	Software-as-a-Service, modern delivery model for software systems
SLA	Service Level Agreement
SOAP	Protocol for exchanging structured information, most commonly used in traditional Web Services
SSL	Secure Sockets Layer: network protocol for secure information transfer
SSL	Secure Socket Layer – security technology for reliable information exchange over the Web
SSO	Single Sign on – access control model for a user to various systems
SVG	Scalable Vector Graphic
TAM	"Technical Architecture Modeling"; modeling standard for architecture diagrams used in the project; TAM is a UML2.0 profile with additional extensions from Fundamental Modeling Concepts (FMC).
TEP	Transport Execution Plan (detailed description of a part of logistics process)
TPM	Transport Planning Module

UI	User Interface, usually referring to graphical user interfaces for human-machine interaction
UML	Unified Modelling Language: established standard for model-driven software engineering published by OMG
VPN	Virtual Private Network (secure & reliable connections for information transfer over the Web)
VPN	Virtual Private Network – security technology for reliable information exchange over the Web
Widget	A part of a GUI that can be added to a web-based Front-End
WSDL	Web Service Description Language (W3C Recommendation)

1. Introduction

As the third Deliverable of WP3 "Solution Design and Technical Architecture", this report provides the interim technical specification of the overall FInest Platform as the continuation of the conceptual design presented in the preceding deliverables of WP3. In addition, we report on the refinements of the overall concept of the FInest Platform and its Core Modules, therewith continuing the alignment between the domain analysis (WP1) and use case analysis (WP2) and the design of the technical solution that is elaborated in WP3 and WP5 – WP8.

The overall aim of the FInest project is to design the future collaboration and integration platform for Transport & Logistics – referred to as the FInest Platform – such that it properly meets the requirements, demands, and expectations of domain stakeholders and end-users. In order to achieve this, the project is continuously refining the overall technical design with early validation and detailed assessment by the domain experts. In the preceding project milestones, we have presented an initial conceptual design under consideration of the early domain and use case requirements (M6), and a refined conceptual architecture along with early demonstrators and a satisfaction study on the detailed domain requirements. At the current milestone (M18), we present the interim technical specification of the FInest Platform as well as refinements of the overall solution design with respect to domain and user expectations as well as the concepts for commercialization. For the final milestone (M24), it is planned to continue the refinement of the overall solution design as well as on the technical specification as the basis for phase II implementation plans, and to provide conceptual prototypes of the Core Modules.

In order to achieve the target results for the current milestone, the project team has set up two parallel but closely alignment work streams:

- A ***Technical Design Stream*** that has been mainly concerned with elaborating the interim technical specifications of the FInest Platform, performing the major step from the conceptual design presented at M12 towards technical specifications as the basis for implementation, including the refined analysis and initial assessment of the Generic Enablers provided by the FIWARE project that appear to be relevant for the technical implementation; this work has been conducted by the technical experts from WP5-WP8 as well as those concerned with the technical building blocks designed in WP3;
- A ***Conceptual Design Stream*** involving both domain and technical experts that has been concerned with the refinement of the overall concept including both:
    - The *refined functional specification* of the FInest Platform from a business perspective, i.e. how it shall work in order to overcome the business challenges that have been identified in WP1 and WP2 during the previous milestones, and
    - The *refined overall conceptual design* with respect to the enablement of ecosystems and the stable and reliable operation of the FInest Platform.

The report presents the results of both work streams in a condense manner; additional documents with the respective details have been prepared and are available to the project consortium. At first, Section 2 presents the results of the conceptual design work stream, including both the refined functional specification and the refined conceptual design. Then,

Section 3 explains the work of the technical design work stream and provides the part of the interim technical specification that is relevant across WP5-WP8, and Section 4 provides the interim technical specifications of the building blocks elaborated in WP3 (incl. the Front-End, the 'Back-End' for system integration, and the Security Framework for the FInest Platform). A substantial part of the technical design work has been the continued analysis and interim assessment of the relevant Generic Enablers provided by the FIWARE project; the results are discussed as part of the technical specifications, and the Appendix provides an overview of the availability of Generic Enablers and the status of investigation and usage by the FInest project. Finally, Section 5 concludes the report along with an outlook to the work planned on the solution design and technical architecture for the final milestone of the project. In

2. FInest Platform – Refined Overall Concept

This Section reports on the refinements of the overall concept of the FInest Platform, focussing in particular on the *functional specification* from a business perspective (i.e. how the FInest Platform shall work in order to overcome the identified business challenges), and the *extended conceptual design* with respect to the commercialization of the FInest Platform.

In order to properly depict the progress on the overall concept and technical design, let us recall the status of the preceding reports. In M12 (Deliverable D3.2), the refined overall concept as shown below in Figure 1 has been presented, along with the identification of the main building blocks (Front-End, Core Modules, Back-End, and Customization Tools), a comprehensive study on the satisfaction of the domain and use case requirements identified in WP1 and WP2, and a set of early demonstrators was developed to showcase how the FInest Platform can facilitate future business scenarios and overcome the deficiencies of existing solutions. For the technical design, *conceptual architectures* (i.e. the definition of features and principle interaction) for the already identified main building blocks were defined, namely: the 4 Core Modules, the Front-End, the Back-End, and the initial Security Framework. These have been taken up by the Technical Design Team in order to elaborate the interim *technical specifications*, which are presented in Sections 3 and 4 and the M18 deliverables of WP5 – WP8. In addition, the overall concept has been refined in to aspects:

1. How the FInest Platform as a whole shall work in order to properly address the domain requirements and demands on future ICT solutions; this is reported in Section 2.1, and
2. The identification, respectively the refinement of additional main building blocks that appear to be relevant for the proper operation and commercialization of the FInest Platform; the results on this are reported in Section 2.2.

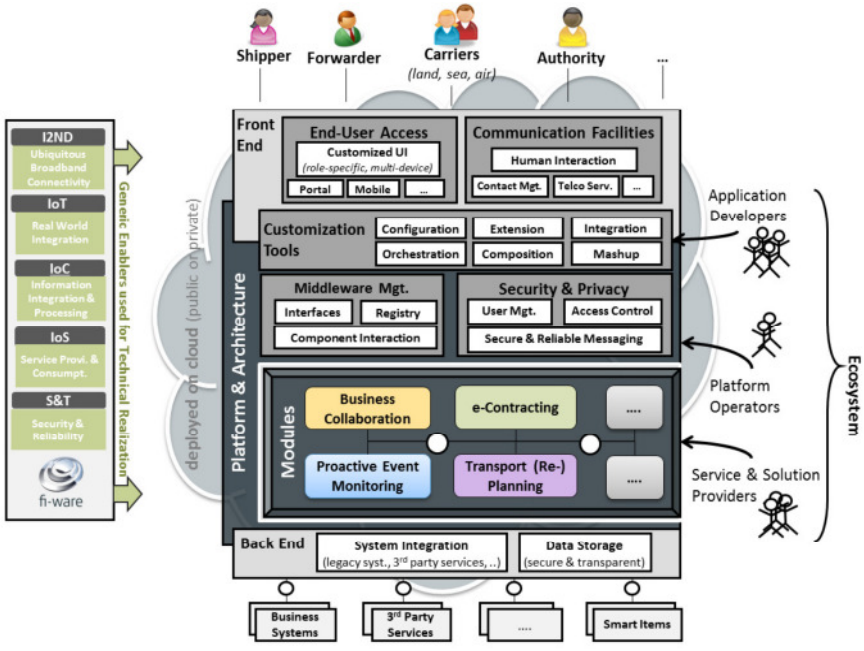

Figure 1 – Overall Concept from M12 (presented in Deliverable D3.2)

2.1. Refined Functional Specification

The first part of the refined overall concept is concerned with clarifying how the FInest Platform shall work in order enable the efficient and effective collaboration in Transport & Logistics with respect to the requirements and expectations that have been identified in the domain and use case analysis. For this, the following first explains the overall operational model, and then depicts the features and usage of the 4 Core Modules designed in the project. Both aspects have been revised since M12 in the course of continuous alignment, feedback, and assessment activities among the Conceptual Design Team.

2.1.1. Overall Operational Concept

The overall purpose of the FInest Platform is to enable the seamless and efficient collaboration of stakeholders involved in the planning, execution, and coordination of transport and logistics processes. For this, the platform consists of three main functional layers as shown in Figure 2 below: the *Front-End* that serves as the main point of access for End-Users (i.e. business experts concerned with managing transport and logistics processes) and provides access to all available features, services, and apps, the *Core Modules* that provide re-usable functionalities to enable real-time business collaboration as well as general and domain-specific business functionalities that can be re-used to rapidly develop customized solutions at minimal costs, and the *Back-End* layer for integrating existing systems (e.g. legacy systems, standard business systems, and external services and systems) in order to allow for continued usage of existing IT landscapes and for exploiting emerging technologies for advanced business features (e.g. IoT-enablement).

The basic operational principle of the FInest Platform is as follows: the Module Layer encompasses components that offer general business and domain-specific functionalities and, by being provided in form of on-demand services with interoperable interfaces, can be re-used and combined for various business purposes. While this layer can be extended with additional modules, the basic functionalities provided by the four Core Modules developed in the course of the FInest Project can be summarized as follows, referring to the more detailed explanations below: the Business Collaboration Module (BCM) maintains a global knowledge base that keeps the information that need to be interchanged among the stakeholders of a logistics process along with the current processing status, and the Event Processing Module (EPM) allows to capture and handle all type of events that trigger or influence the progress of a logistics process; together, these modules enable that all information can provided to each stakeholder in real-time in an event-driven manner. On this basis, various domain-specific functionalities can be developed for standard tasks in transport and logistics managements, such as e.g. order management & tendering, transport planning, and monitoring & tracking during the execution. The Transport Planning Module (TPM) can be considered as an example for such domain-specific services: it offers transport route planning facilities with up-to-date information from the BCM and the ECM, and can be re-used in various scenarios and applications that require such planning facilities. Finally, the E-Contracting Module (ECM) provides the basic features to offer and find the business services provided by Logistic Service Providers and manage the contracts between business partners, therewith allow stakeholders in transport & logistics to act in open business networks via e-marketplaces.

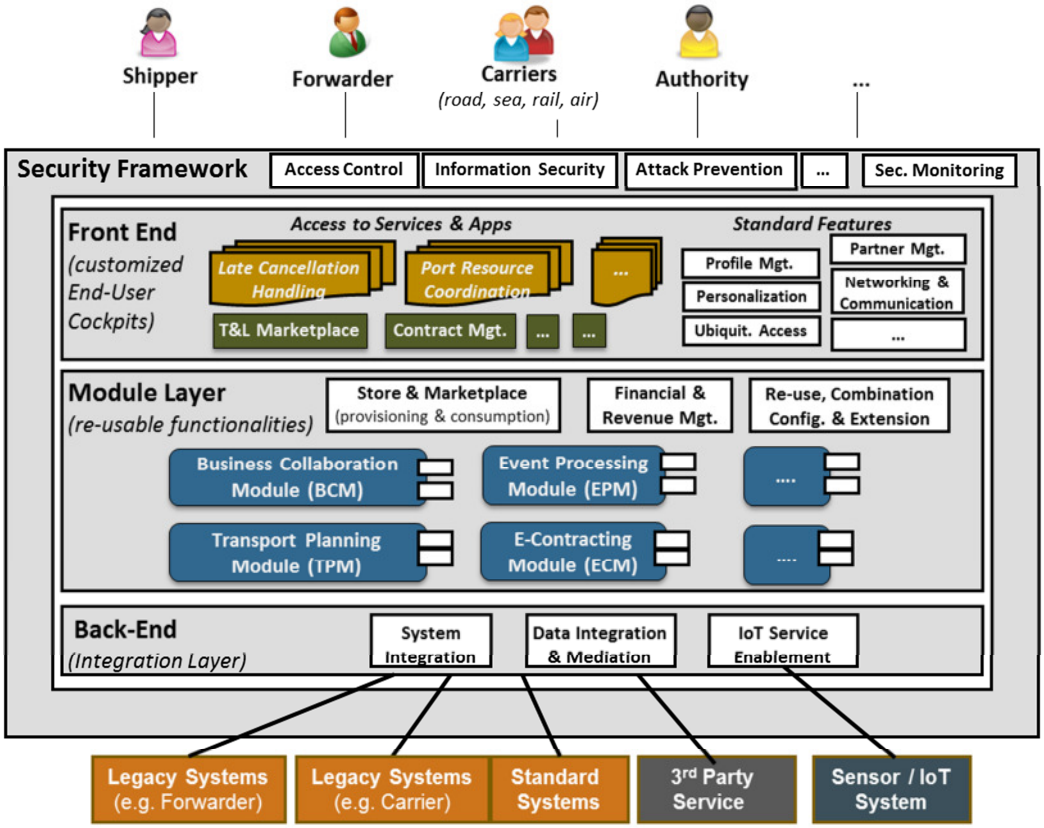

Figure 2 – Refined Overall Operation Concept

Based on these re-usable components, specific applications can be developed in a rapid manner by re-using and combining the functionalities provided by the Core Modules. As an example, consider the applications for the Demonstrator 'Real-time Event Handling' that allow carriers and forwarders to handle updates to transport orders on a short notice (see Deliverable D6.2 for the details): for this, the EPM is used to capture and handle the event 'order update by shipper', the BCM to immediately notify the involved forwarder and carriers, and the TPM to conduct re-planning for the logistics process in quick, automated manner, for which the ECM is used to find relevant offers of transport providers; all of the Demonstrators presented in the M12 deliverables work analogously.

All of these solutions – referred to as Apps, as the intended user experience and economic models are similar to today's apps for smart phones – are accessible via the Front-End. This also includes 'standard features' for e.g. profile management and personalization, partner and community management, and social networking and collaboration features, therewith allowing for a 'all you need in one place' user experience; the refined design and interim technical specification is presented below in Section 4.1. Via the techniques for system and data integration provided in the Back-End layer, the existing IT landscapes used by End-Users (resp. their organizations) as well as external services can be connected to the FInest Platform, so that information can be im- and exported (see Section 4.2 for the interim technical specification). Finally, the Security Framework encompasses the relevant techniques for ensuring access control and secure information transfer, handling, and storage, therewith ensuring that the FInest Platform is 'secure by design' (see Section 4.3 for the interim technical specification).

2.1.2. FInest Core Modules – Features & Functionalities

The following outlines the main functionalities that are provided by the four Core Modules that designed in the course of the FInest project, with respect to their for business-relevant features and re-usability. Further details, in form of interim technical specifications, are provided below in Section 3 (technical interaction and interfaces), and in the Deliverables of WP5 – WP8.

2.1.2.1. Business Collaboration Module (BCM)

The BCM establishes a global knowledge base for collaborative business processes. For each business process the BCM tracks, it holds a global data model which encompasses all information about the process. Based on a transport plan, which is the outcome of a planning process and not part of the BCM, a data model is initialized with a transport's core data. The BCM allows users to enter additional information, such as waybills, crew lists or customs documents. All information is visible to other users involved in process, if they have the necessary access rights. Thus, the BCM manages all information of transport processes and make it available to its user in real-time.

Additionally, the BCM is also in charge of keeping the contained information up-to-date. During the execution phase of a business process, the BCM gathers external events and integrates these into the internal data model in real-time. By this, the model is kept up-to-date so that it reflects the real-world situation at all time. By accessing the BCM, stakeholders are able to get all updated information about their current business processes at any time and can immediately react to deviations that may occur. Furthermore, through the central storage of business data, which is usually scattered over different stakeholders of a business process, the BCM is able to establish a global process control and inform users about issues during the process execution at the time of occurrence. To enable all this, the BCM uses a novel data model, which is able to tightly integrate data and process information and by this make it easy to integrate data from different sources as well as natures. The modelling approach is based upon the entity-centric modelling and the BCM implements this approach for the basic building blocks of the internal data model, the Collaboration Objects.

In FInest we focus on a very specific type of collaborative business process: logistic processes involving multiple business partners in order to transport goods. Although, the basic functionality of the BCM, as described above, is not specific to a certain domain, the internally used model is specific indeed. Thus, the BCM has to provide a specific data model for transport processes and we developed a model that builds upon the 'Common Framework' of the eFreight project[1] with some extensions. The 'Common Framework' introduces a set of data (meta-)models which can be used to describe logistics process and surrounding activities such as tendering and booking. Each (meta-)model defines the structure and meaning of data fields. This enables a computer-based processing of the contained information. For each new transport process the internal data model of the BCM is initialized by a transport plan, the outcome of the TPM. This plan provides the basic parameters of the process an enable the BCM to register itself for necessary events which are needed to track the process. The model behind the transport plan is also based on the Common Framework and so an easy alignment is ensured. Due to the

[1] http://www.efreightproject.eu/

differences between types of diverse transport processes, the BCM does not use a fixed data model but provide flexibility based on a skeleton and add-in concept. A specialized interface allows users to access the data stored within the Collaboration Objects and by this, be able to get the information that is necessary to get global view on observed transport process.

Summarizing, the main capabilities of the BCM Module are:

- Captures information that must be exchanged among involved stakeholders as well as the expected process steps in form of co-called 'Collaboration Objects'
- Current Collaboration Objects based on 'Common Framework' from e-Freight with some extensions
- Provides a flexible structure of the data model in order to support different kinds of transport processes
- Constantly updates the internal data model during the execution of transport processes
- Provides interfaces to query the internal data model (to enable re-use of information kept within Collaboration Objects for other domain-specific functionalities).

2.1.2.2. Event Processing Module (EPM)

The EPM is the central observer of external and internal event sources and the supplier for events, which are needed by the BCM in order to keep its internal model up-to-date. In the beginning of a transport process, the EPM receives parameters about the transport process which indicate what kind of events are important for that process and have to be tracked. The BCM provides the ECM with this data and it is based on information from transport plan (TPM) as well as the BCM's internal data model. The actual event processing is based on a set of event processing rules and the EPM uses the provided parameters to activate the necessary rules. If necessary, the rules also will be parameterized based on the received data from the BCM.

During the execution of a transport the EPM receives events through its various interfaces to external or internal systems. This could be the FInest user interface, if a user adds a specific document or acknowledge the reception of cargo for example, as well as sensor networks or other FInest modules. Through the previous processing rule selection the EPM is able to filter the important events for particular transport processes and notifies the BCM about their occurrence. The EPM is also able to act pro-actively. This means that is able to detect the absence or receipt of certain event beforehand. If a storm forces a harbour or airport close for example, the EPM is able to determine that the cargo handled at this places will not be able to be dispatched (depending on the time of the storm) and send events direct to the user before the deviation in his or her transport even occur.

Summarizing, the main capabilities of the EPM Module are:

- Support for user-events (manual inputs) and automated events (e.g. from a sensor network)
- Pro-active event management (e.g., providing forecast features, etc.)
- Interfaces to capture events (e.g. from User Front-End or a from connected legacy / sensor system) and 'event notification' for other modules (e.g. for triggering automated status updates for the events.

2.1.2.3. Transport Planning Module (TPM)

The TPM is the initializer of a transport process. For a given user demand, the module creates a transport plan which then serves as basis for all subsequent steps. This transport plan contains all information necessary for the execution phase. It is used by the BCM, for example, to initialize the internal Collaboration Object-based data model or by the EPM to activate and parameterize the necessary set of event rule in order to be able to receive the correct events during the execution of the transport. The transport plan is based upon the 'Common Framework' from the eFreight Project, whereas two document types (meta-models) are of high importance for the transport plan:

- Transport Service Description (TSD): Describes a transport service and make this discoverable by algorithms. The service descriptions are used by the TPM in order to find suitable service providers for a certain segment within the requested transport
- Transport Execution Plan (TEP): The outcome of the (re-)planning process is set of TEP. Each of these documents describes a bi-lateral relationship between a transport service provider and its client. Through the aggregate of these documents the whole transport process is described and all necessary information for the execution is incorporated within this document set.

In order to manage the multiple TEPs the TPM uses a third kind of document, the Transport Chain Plan (TCP). This document type defines the relationship between TEP documents and is not defined by the Common Framework.

In order to fulfil its (re-)planning task the TPM tightly interacts with the ECM in order to meet the requirements of the user at minimal costs. This is done by the consideration of established contracts between the user and his or her transport service providers, whereas the contractual relationship is managed by the ECM. By this interaction, the (re-)planning process takes existing contracts of the requesting user into account and is able to find the best suitable transport routes.

In addition to this, TPM also provides some advanced features. It supports order management of end-users, provides algorithms to optimize transport routes and allows a constant re-planning of (on-going) transport processes in order to react to deviations.

Summarizing, the main capabilities of the TPM Module are:

- Find transport routes by considering available offers and/or existing contracts (esp. using ECM to find suitable transport services offered by known and unknown LSPs)
- Support for transport ordering
- Optimization of routes
- Allow re-planning (e.g., change of route due to some delay)
- Interfaces to 'export' created transport plans (e.g., used by the BCM) and interfaces to use other modules (e.g. get transport service & contract information from ECM).

2.1.2.4. E-Contracting Module (ECM)

The ECM is an open e-marketplace for the T&L domain. It connects itself to various existing logistic marketplaces and allows the user to search these via a single access point. This is beneficial for service provider, who can trade their services over different marketplaces at once, and the service clients, who can search for suitable offers at only one place.

But the ECM does not merely aggregate the access to different marketplaces: it also provides an ample tool set for contract management. Users can enter their existing contacts to service clients or providers and let the ECM manage it. The conditions of each contract were updated by the BCM after the execution of a transport process in order to have an up-to-date view on the used resources ensured by a contract. During the planning phase of a transport, the TPM uses information about the managed contracts in order to find the most suitable transport route. Existing contracts are prioritized or if the user demands it, new contracts can be established. With this the ECM is the central component to establish and manage bi-lateral business relationships between partners. Its interfaces to external components allow an effective use of contract information and keeping the contracts up-to-date.

Summarizing, the main capabilities of the ECM Module are:

- Search functionalities for the business services offered by LSPs and other stakeholders
- Management of bi-lateral contracts between business partners
- Interfaces to allow automated access to the 'e-marketplace'.

2.2. Extended Conceptual Design

The second part of the refined overall concept is concerned with identification of additional technical building blocks that appear to be needed for the FInest Platform, in particular to allow for the envisioned commercialization and the proper operation and usability. For this, the following 3 elements have been identified:

- A Store that allows End-Users to consume the services and apps that are provided by IT solution providers, including the financial management of the FInest Platform
- Technologies for middleware and component management to allow the proper operation of the FInest Platform, and
- Techniques for the customization of services, apps, and solutions to the individual business needs of End-users.

For each of these, the following explains the business needs, outlines the envisioned solution approach, and identifies the Generic Enablers that appear to be relevant for technical realization.

2.2.1. Store for Services and Apps

During the refinement of the overall conceptual design of the FInest Platform, it became obvious that an important building block is missing in order to allow for the commercialization of a collaboration platform. We defined that each module of the platform has to provide service-based interfaces in order to facilitate extension and adaptation. A middleware layer shall allow

the individual combination of provided services and therewith, also the easy integration of external or third-party services[2] (see Section 4.2 for further information about the integration of external ICT solutions). However, the platform was lacking a facility to enable end-users to easily select external services and integrate them within their own FInest Platform environment[3]. Hence, we introduced the concept of a Store that will make apps or services from external developers available to end-users. The store allows developers to publish their apps and defines requirements which have to be fulfilled. Each app can make use of the functionality and data that is managed by the FInest platform. This allows external developers to build upon the solution FInest provides and introduce specialized solution for very particular use cases. If an end-user needs additional functionality for a specific use case, he or she can browse the App Store in order to find suitable apps and integrate them per click. The FInest platform is taking care of the integration. As common in other App Stores (e.g., Google Play), the revenue an app is able to gain will be shared between the developer and the owner of the FInest platform. In case of the FInest App Store this could mean a fixed rate and additional charges if features of the platform are used.

In summary the App Store shall consist of the following features:

- *App Repository*: The repository where all registered cSpace Apps are available (most likely based on (Linked) USDL descriptions)
- *App Discovery Support:* Tool Support to allow Consumers to search, find, and inspect available apps
- *Purchase of Apps:* Tool Support for allowing Consumers to 'buy' apps
- *Discovery & Re-use Support for Developers*: Tool Support to allow Developers to search, find and inspect available apps to develop new apps
- *Provisioning Support for Developers*: Tool Support to allow Developers to publish Apps in the store
- *Financial & Revenue Sharing Management*: Manage financials of the App Store (income by end-user payments, revenue sharing among providers)

For the latter, the works on the value added network analysis for the FInest Platform shall serve as a basis. An initial version of the role identification and the economic structure of the ecosystem that shall be enabled by the FInest Platform has already been presented in the previous report (see D3.2, Section 2.2). This has been refined in close interaction with the domain experts from WP1, and extended with a detailed analysis of possible pricing models along with an initial economic feasibility study based on interviews with the use case partners form WP2. This work is however still in progress, and hence will be reported in detail at a later point in time.

[2] Note: The technical integration of external ICT systems is not part of the Store for Services and Apps. Offered Services or Apps can make use of Backend functionality to communicate with external systems. The usage of this functionality is part of the Services and Apps and not of the Store.
[3] Since the technical realization of user-specific services or apps is not elaborated yet, we used the term "environment" to abstract from the technical details.

Regarding the technical realization, our investigation of the GEs provided by the FIWARE Apps[4] chapter showed that these would provide a comprehensive basis to realize the FInest App Store. Based on the Marketplace GE and Store GE FInest can build its own App Store with tradable services and apps. A discovery support can be realized by the means of the Registry and Repository GE. Additionally, FIWARE GE provides for revenue sharing that can be used to enable different sharing models as foreseen. The most of the described GEs depend on Linked-USDL language to describe the contained services and apps. This also fits our requirements since we already have experience with the modelling in Linked USDL since the ECM use it for transport contract modelling. In summary the relevant GEs are:

- Apps.Repository GE
- Apps.Registry GE
- Apps.Marketplace GE (incl. Discovery & Matchmaking Component)
- Apps.RSS GE
- Apps.Store GE

2.2.2. Platform Middleware & Management

One of the technical architecture considerations is to provide means to allow an efficient and sufficient mechanism for FInest modules to interoperate as required by services and applications to be developed on top of the FInest Platform. Another consideration is the provisioning of means to properly deploy, manage and sustain the modules and services on a cloud infrastructure supporting a reliable, elastic and sustainable execution of FInest.

In FIWARE vision and plans there are considerations for generic enablers to support these requirements from FInest, such as the additional middleware generic enablers solicited for in FIWARE's 1st Open Call as well as various generic enables to support Platform as a Service (PaaS) in the cloud for which only high level descriptions exist. Both of the mentioned intentions are not expected to be completed by the completion of this delivery. Hopefully for the next deliverable D3.4 regarding implementation plan for Phase 2, these intentions could be properly evaluated and considered for supporting the FInest middleware requirements.

It is also important to note that the middleware approach may be heavily dependent on an underlying cloud infrastructure and may even require appropriate packaging of the FInest modules and developed services and apps to be deployable and manageable through the middleware by the selected underlying cloud infrastructure.

For the above two reasons, FIWARE schedule to elaborate on the generic enablers that could support the middleware requirements and the dependency on a selected underlying cloud infrastructure, the completion of the prototypes in this project will include specifications and implementations of one-to-one, module-to-module interactions by defining APIs that any two modules that need to interact agree on the method and payload that enables that interaction, as seen in the TAM models to follow.

[4] Applications and Services Ecosystem and Delivery Framework

2.2.3. Customization, Extension, and Personalization

The transport and logistics domain is coined by a wide range of stakeholders and actors with very different business interests, who all need to interact in a variety of different scenarios. This starts from various requirements for different means of transports, goes on to the vast amount of different regulations from country to country and maybe ends at the different process implementations of specific companies. A platform that is supposed to bring all these different stakeholders together cannot support each scenario out of the box. Thus, it has to provide a way to be customized, extended or personalized.

The FInest platform takes this in account since the very beginning of its design. Modules of the platform are designed modular and through the use of common interfaces and standards modules can be exchanged or integrated. However, this only supports the adaptation of the platform level and is not flexible enough if a FInest instance shall address a wide range of people. Hence, other approaches for customization have to be provided.

The basis for the customization and extension approach for the platform is established by the components we described within the last two sections: the App Store and the Middleware Management. Whereas the first provides the possibility to easily add or exchange functionalities for specific users (or user groups), the latter ensures that the different services and apps will be able to work hand in hand. To enable this and facilitate the development of apps, FInest will provide a toolkit for app developers. This ensures the developed apps comply with the technical governance from FInest on the one hand and provides developers with an easy access to the standard features of the FInest platform on the other hand (e.g., services, data models, etc.). For the realization of the toolkit we currently consider an Eclipse-based environment with FInest specific plug-ins and wizards. Furthermore, we plan to provide specific tools for the creation, configuration and extension of Collaboration Objects as well as event handling rules. The toolkit shall also provide support for system and data integration from the backend (see Section 4.2 for more details about the integration of external ICT systems) and facilitate the orchestration as well as mash-up of Apps.

Based on the availability of external and specialized solutions, FInest allows IT experts to pre-configure different platform accounts on the level of organizations, business units and specific units. By this, a company can select a set of FInest and external services which shall be used by each customer and therewith, adapt to their very own needs. Services can be orchestrated, connected to external systems and mash-upped beforehand, so that a single user has not to deal with this. Furthermore, the used UI concept of FInest enables the personalization of the Front-End UI for each user. See Section 4.1 for details.

3. Overall Interim Technical Architecture

This section presents the interim technical architecture of the overall FInest Platform, i.e. the detailed technical design that shall serve as a profound basis for the technical realization and presents the main results of the Technical Design work stream. As outlined above, for this the detailed technical specifications of those technical building blocks that were already identified at the M12 milestone have been elaborated, including the refined analysis and initial assessment of the Generic Enablers provided by the FIWARE project. Therewith, the interim technical architecture constitutes the first refinement of the conceptual architecture developed in the first year of the project towards a technical solution. Following the iterative methodology applied throughout the FInest project, this will be extended and refined into the technical specification and an implementation plan to be delivered in M24 of the project.

The section is organized as follows: Section 3.1 explains the overall process and methodology applied in the FInest project for elaborating the technical design and specifications. In Section 3.2 we first briefly recap the conceptual architecture as of M12, as this has been used as starting point for the technical design activities. This conceptual architecture also includes smaller updates applied during the technical design work to incorporate findings and insights gained while developing the technical architecture. In Section 3.3 we then provide a high-level view on the interim technical architecture. This provides a basic breakdown into technical components (Section 3.3) as well as technical interfaces between these (Section 3.4). The interim technical specifications are complemented by those of the four Core Modules that are presented in deliverables D5.3 – D8.3, and the revised technical design and interim specifications of the Front-End and Back-End layers as well as for the Security Framework of the FInest Platform that are presented in the next section (see Section 4).

3.1. Technical Design Approach

The design work in FInest follows an incremental and iterative design process, which allows for incorporating novel findings and developments. The process has three major defined "milestones":

- **M12 – FInest conceptual design:** The conceptual design (aka. **conceptual architecture**) describes the logical structure of the FInest solution (in terms of user-visible features and feature-units, aka. modules), as well as the principle interactions between those modules.
- **M18 – FInest interim technical specification:** The interim technical specification is provided in the form of an **initial technical architecture**, which provides a first breakdown into technical components as well as technical interfaces between those components. In simple terms these technical components and interfaces realize the conceptual features and interactions as laid out in the conceptual architecture.
- **M24 – FInest technical specification and implementation plan:** The finest technical specification will elaborate the technical architecture by providing a further level of refinement, down to the level of programmatic APIs. Whereas the intermediate

technical architecture is expressed by means of design models, the final technical specification will drill down to the programmatic implementation of the FInest solution.

This deliverable focuses on the results of "milestone" Mo 18. Accordingly, we will present the design approach to reach this milestone in the project by introducing the key technical design decisions the FInest design team had to take (Section 3.1.1), the languages and guidelines used to define the technical architecture (Section 3.1.2), as well as the organization and facilitation of the design work in general (Section 3.1.3).

3.1.1. Technical Design Decisions

The definition of the FInest technical architecture started from the conceptual architecture delivered in M12 (see deliverable D3.2 and Section 3.2). In order to refine this towards a technical solution, the following key design decisions were taken by the FInest design team:

Refinement of modules into components:

For each of the logical modules, an internal software-structure has to be designed, taking into account a suitable separation of concerns, as well as the potential use of FI-WARE Generic Enablers. To introduce FI-WARE component models into our design documents, we performed a reverse engineering step, during which we started from the FI-WARE specification (which was too low-level, i.e., on the level of RESTful interfaces, for our purposes). In fact, this approach was inspired by the InstantMobility project, which performed a similar exercise during its design work.

Definition of technical interfaces:

For each of the components, the operations offered were defined (i.e., in terms of functionality/functions or data/information offered or required). The offered operations were then clustered into meaningful interfaces. The design decisions that needed to be taken here were to find manageable clusters of operations, while still maintaining a granularity that would allow the interfaces to be used / reused for different purposes. In a sense, those interfaces can be considered atomic services that may be offered by the FInest modules.

Definition of data types:

To precisely describe the input and return parameters of the operations offered by the technical interfaces, data types where defined in cases where standard (i.e., simple) data types were not sufficient.

3.1.2. Modeling Language and Guidelines

As mentioned above, the intermediate technical architecture is defined at the model-level, to provide a concise and abstract description of the FInest design, allow for discussion, consolidation, as well as user feedback. To this end, the design team agreed on using TAM (the Technical Architecture Modeling language), a UML derivate, already used for modeling the conceptual architecture in D3.2. Together with TAM, the design tool of choice was MS Visio to

exchange and integrate the results of the design team members, as TAM Stencils were readily available for the tool[5].

Along the above three key design decisions, three kinds of diagrams have been defined as expected outcomes of the design work:

Component definition:

In order to represent the technical components and interfaces of the FInest platform, component diagrams of TAM are employed. This diagram type allows for the hierarchical definition of components, where each component can specify provided and required capabilities by defining interfaces (using the "lollypop" notation).

Below a very simple example of such a diagram is shown, depicting a module, which is decomposed into three components, connected by three interfaces. In addition, the module offers specific capabilities through an interface (*IOffered*), and one of its components requires the capabilities of an external component (*INeeds*).

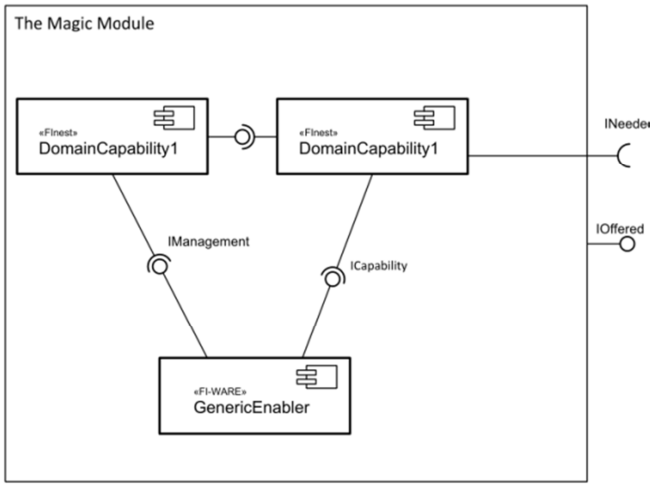

Figure 3 - Example Component Diagram

In addition, we use the UML stereotype mechanism to distinguish between FInest specific components (<<FInest>>) and generic components offered in the form of FI-WARE Generic Enablers (<<FI-WARE>>). As mentioned above, the FI-WARE component models have been identified through a reverse engineering step from the FI-WARE specification (which was too low-level, i.e., on the level of RESTful interfaces, for our purposes).

[5] It should be noted that the Stencils did not strongly enforce consistent use of model elements (such as connectors, containers, etc.), thus slightly different visual representations can be observed in the designs. Despite the fact that we do not perceive these notational inconsistencies critical to the understanding of the architecture, the model representation is planned to be made consistent for the final version of the architecture delivered in Mo24.

Interface definition:

Interfaces in TAM are described using class diagrams. The operations of the Interface are defined in the respective compartment of the class model (see figure below).

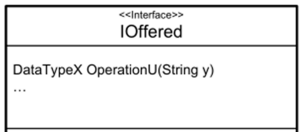

Figure 4 - Example Interface Definition

Where feasible, those visual models are replaced and/or complemented by a tabular description of the interfaces.

Data Type definition:

Definition of data types is done using class diagrams, stereotyped with <<dataType>>. An example can be found below.

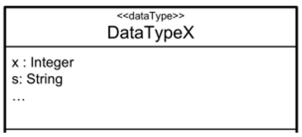

Figure 5 - Example Data Type Definition

Again, where feasible, those visual models are replaced and/or complemented by a tabular description of the interfaces.

3.1.3. Design Facilitation

To ensure targeted progress concerning the design work, bi-weekly physical/virtual meetings of the FInest design team have been performed, facilitated and chaired by the FInest technical coordinator (UDE). All design team members presented their progress wrt. the design of the modules they were responsible for and the team discussed open issues and drove the overall design to a coherent and consistent architecture as far as possible. Due to the complexity and innovative nature of the FInest platform, it should be understood that what is presented by M18 of the project can only constitute an intermediate result of this design work. The design team will continue to improve, consolidate and refine the design to deliver the planned technical specification and implementation plan by Mo24 of the project.

3.2. Updated High-level Conceptual Architecture

The following recaps the updated conceptual architecture of the FInest platform, which as initially been presented in D3.2 and presented in [1], [2].

As shown in Figure 6, the FInest platform is structured into three layers. The FInest platform itself is realized using service-oriented and cloud technology, facilitating interoperability,

openness and extensibility through standard interfaces. In addition, the use of integrated security and privacy management mechanisms ensures the secure and reliable exchange of confidential and business-critical information; including mechanisms such as access and identify management, artifact-based security control, and secure storage & backup.

Front End Layer:

The front end layer of the FInest platform provides users with role specific, secure, ubiquitous access from different devices to information concerning the operation of the transport and logistics network. The capabilities offered by the FInest platform and its core modules will be offered through a customizable "web" portal. Each user can configure this portal by selecting dedicated "apps" depending on the capabilities needed to perform a user's respective tasks – quite similar to the iGoogle or iPhone/iPad model. In addition, the front-end will be integrated with messaging systems (such as SMS, E-Mail and Social Networking) such as to notify users and trigger actions.

Back End Layer:

The back end layer of the FInest platform provides access to, and integration with, legacy systems, third-party services (Internet of Services) and Internet of Things (IoT) devices. Specifically, the IoT devices will provide (near) real-time information concerning the transport processes as well as their context, thus allowing to quickly and proactively responding. Legacy system integration is facilitated by service-oriented technology, e.g., by exposing features of legacy systems as Web services.

Modules Layer:

The modules layer of the FInest platform allows plugging in targeted transport and logistics service modules. Those modules – in the future – may be offered by third parties (such as small and medium enterprises). The initial release of the FInest platform will feature four open-source modules (called Core Modules) for contracting, planning, monitoring and execution of trans-ports.

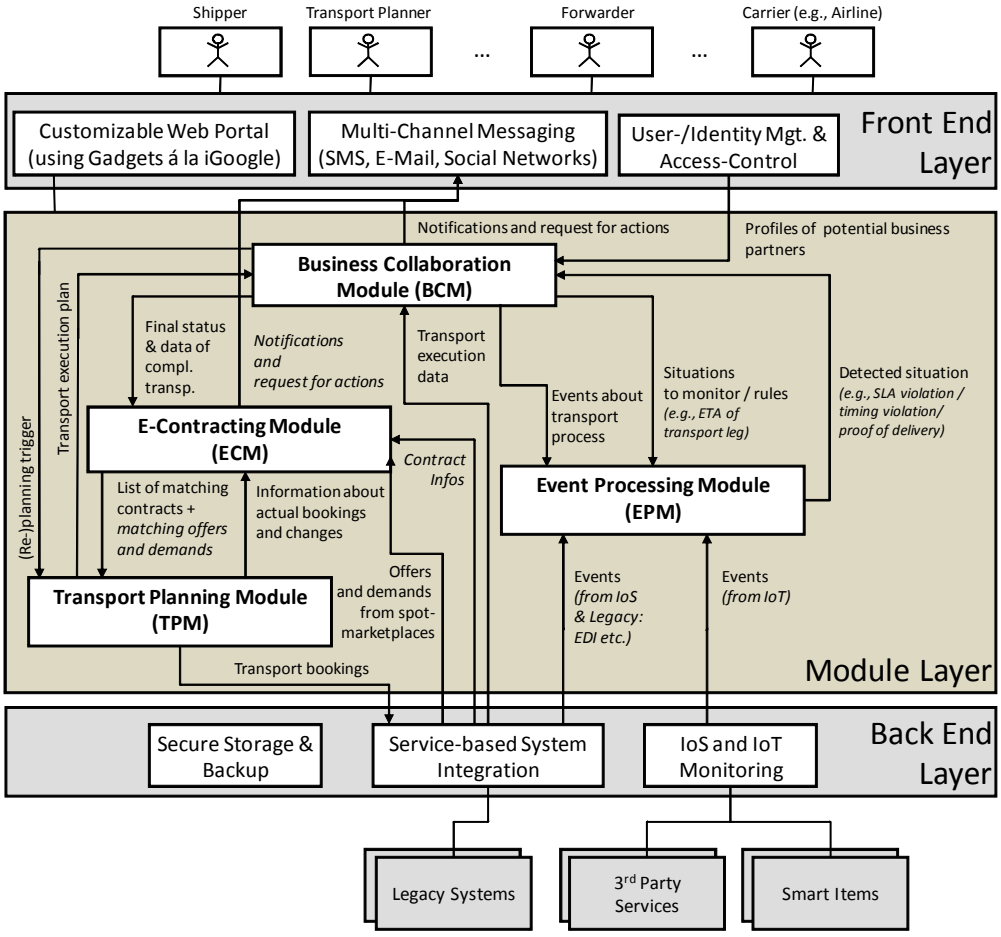

Figure 6 – High-Level Conceptual Architecture of the FInest Platform (M12, see D.3.2)

Please note that (following the iterative design approach of FInest) in parallel to the technical specification; we have updated the conceptual design to incorporate our additional findings and new insights during the design work. The updated / new interfaces are depicted in *italics* in Figure 6.

3.3. Initial High-level Technical Architecture

Figure 7 shows the high-level TAM model of the intermediate technical architecture of the FInest platform, focusing on the design of its core modules. The design of the front-end and back-end layer is elaborated in Section 4 together with other main building blocks of the FInest solution.

Please note that, in order to keep this section concise, the decomposition of the coarse-grained components into more detailed technical software elements, as well as the usage of FI-WARE GEs, are depicted in the technical modules deliverables D5.3 – D8.3.

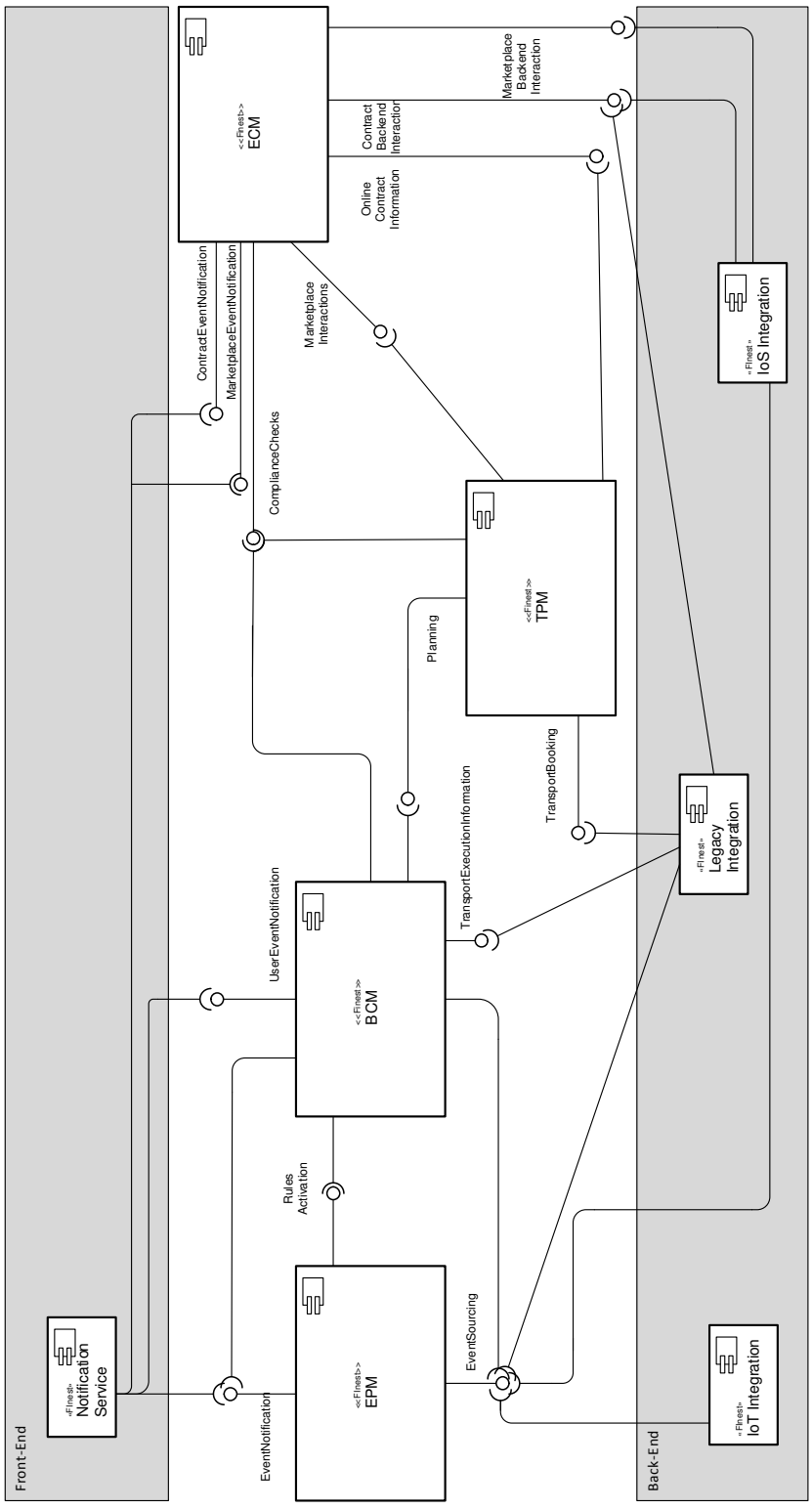

Figure 7 - Intermediate Technical Architecture of the FInest Platform

Note: the front- and backend-end layers only show elements with specific connections to core modules; those are elaborated in Section 4 below

3.4. Technical Interface Definitions (M18 Results)

This section provides an overview of the technical interfaces provided by the Core Modules and layers of the FInest platform. For each interface, we describe its general intent and capabilities, as well as how these are employed to realize the conceptual interactions as defined in the conceptual architecture in Section 3.1.

Table 1 – Technical Interfaces of FInest Platform (Core Modules)

Interface Name	Summary	Provided by	Realized Conceptual Interactions (*cf. Figure 7*)[6]
EventSourcing	Allows connecting event sources (e.g., pub/sub) to the event processing facilities	EPM	• Events (from IoS & Legacy: EDI etc.) • Events (from IoT) • Events about transport process
RulesActivation[7]	Allows activating pre-defined event patterns for a specific transport instance	EPM	• Situations to monitor / rules (e.g., ETA of transport leg)
EventNotification*	Allows being notified about identified complex events	EPM	• Detected situation (e.g., SLA violation / timing violation/ proof of delivery)
TransportExecutionData	Provides information about completed transports, such as to allow for, e.g., managing contracts	BCM	• Final status & data of completed transport process
UserEventNotification*	Provides events about user-relevant transitions in the business process	BCM	• Notifications and request for actions
TransportExecutionInformation	Allows accessing transport execution data where not already provided by other sources	BCM	• Transport execution data
OnlineContractInformation	Allows entering and retrieving contract data during operation	ECM	• List of matching contracts (long-/short-term)
MarketplaceInteractions	Publish logistics service offerings and demands via marketplaces (connected to FInest)	ECM	• *List of matching offers and demands*

[6] *Italic* entries are links from the updated conceptual architecture (see Section **Error! Reference source not found.**)

[7] Please note that the definition of event types (i.e., ECA rules) is assumed to be done manually. Tool / IT support is planned for Phase II (see D6.3)

ComplianceChecks	Provides for various contract-related checks (such as booking against contract / execution against booking)	ECM	• Information about actual bookings and changes
ContractEventNotification*	Notifies about user-relevant events / changes of contracts	ECM	• *Notifications and request for actions*
MarketplaceEventNotification*	Notifies about user-relevant events / changes in the marketplaces	ECM	• *Notifications and request for actions*
MarketplaceBackendIntegration	Access to transport service marketplaces (external to FInest)	ECM	• Offers and demands from spot-marketplaces
ContractBackendIntegration	Access to legacy/existing contract management systems	ECM	• *Contract Infos*
TransportPlanning	Plan and re-plan transport execution chains	TPM	• (Re-)planning trigger • Transport execution plan
TransportBooking	Perform actual booking of service through legacy / existing services	TPM	• Transport bookings
User Profile Information	Allows accessing user profiles to establish B2B connections	Offered by Front-end	• Profiles of potential business partners

Note that the interfaces EventNotification, ContractEventNotification and UserEventNotification are intended to be a specialization of a generic interface GenericEventNotification, providing key data and notification mechanisms for events.

4. Main Building Blocks (Interim Technical Specification)

In this section we address the technical specification of the following main building blocks of the FInest Platform for which conceptual architectures and requirements have already been defined in the preceding deliverables (esp. D3.2, Section 4):

- **Front-End:** realizes the Front-End layer with End-User Access and Communication Facilities
- **System & Data Integration:** realizes the System Integration element of the Back-End layer
- **Security, Privacy & Trust Framework**: realizes the security framework in order to ensure access control and information security on the FInest Platform.

The elaboration of the interim technical specifications has been conducted by the Technical Design Team, following the methodology and work plan as described in Section 3.1 above. The following sections present the interim technical specification for each building block, covering the design considerations, the interim technical architecture specifications using TAM (the Technical Architecture Modeling language, see above), and report on the interim assessment of the Generic Enablers from FIWARE that have been identified to be relevant for the technical realization. In addition, refinements on the conceptual design are provided, e.g. for the Front-End building block to encompass facilities for customization by End-Users.

4.1. Front-End

In this section we will discuss the technical design of the FInest Front-End in order to give a comprehensive overview about the taken approach as well as its intended implementation. For this we will recap the vision for the Front-End in a first section and present the baseline technologies afterwards. Based on this, we explain the technical architecture of the Front-End and evaluate relevant FIWARE Generic Enablers.

The Front-End shall be the ‘web-based portal’ that provides the single point of access via which End-Users access and work with the Services & Apps facilitated by the Core Modules. All necessary information is accessible over it and therewith, an ‘all-you-need-in-1-place’ user experience is established. Furthermore, all interactions between business partners can be conducted over the Front-End, which encompass business transactions as well as direct (unstructured) communication between users.

The refined conceptual architecture of the FInest platform (*cf.* Deliverable D3.2 – Section 4) classifies the Front-End consisting of three different components:

- Multi-Channel Messaging

- User management and access-control
- Customizable Web Portal

Each of these conceptual components is represented in the technical design of the Front-End and will be discussed in detail in Section 4.1.3. But before that we will discuss the overall design considerations for the Front-End.

4.1.1. Customizable Web Portal

In Deliverable D3.2 (*cf.* Section 2.1.1), the User Interface (UI) of the Front-End were described as a web-based portal, which serves as a basis to integrate different functional components. These components were denoted as Widgets, whereas the term does not refer to traditional widgets used for desktop-based user interface development (e.g. buttons, tables or toolbars). We used the term in accordance to its meaning in web-related domains. There this term refers to a small, but fully-fletched, client-side application authored using common web technologies as HTML or SVG. The benefit of Widget is the encapsulation of functionality, which enables a detached usage. Users can combine different Widgets and integrate them in their portal based on their very own demands. Hence, the overall UI for an application is defined by the sum of all Widgets and hence, our idea is comparable to other big Widget-based web portals, like *iGoogle*. Consequently, the portal is acting as a framework to manage and incorporate Widgets depending on the selection of a user. But it will also provide common features, which are independent from a specific use case. Examples for FInest are user profile definition, management of business contacts or configuration interfaces.

Initially, Widgets will be provided by the Core Modules. Each module provides a set of widgets to make their features available for end-users. In a second step, user or third party providers can to start to develop their own Widgets and offer them via a shop to other users. With this approach also very specific usage scenarios can be supported, which is of high importance for the highly dynamic T&L domain. This also enables the customization of the Front-End. Users are provided with an easy way to include customized code in the standard UI of FInest. Different external services can be used in order to support not directly addressed use cases. Hence, custom Widgets smoothing the way to a customizable web portal.

4.1.2. Technical Realization

From a technical perspective, most Widgets consist of HTML+JavaScript to describe and program their functionality, a standardized packaging format and metadata. Hence, every Widget can be conceived as a single Web page (or rather Web application) with their own DOM model. Specialized Web server applications – so called Widget Container – use the metadata to manage the Widgets and make them available for users. The metadata usually encompasses fields to describe the name of a widget, its author or description, but can also be used to make a Widget tradable with the necessary information is included. The packaging format defines the structure of the Widget and its resources; additionally, it a model for the metadata. Currently,

there a two widely accepted and open formats available: Widget Packaging and XML Configuration[8] (short: W3C Widget) and OpenSocial[9].

W3C Widgets are a W3C recommendation from 2011. It encompasses a specification for a packaging format and the accompanying metadata document. The packaging format relies on PKWare's Zip specification as archive format and defines the structure inside it. The metadata document is XML-based and encompasses general information about the Widget (such as name, description or author – metadata) or configuration parameters used by the container.

Developed and pushed by Google, OpenSocial raised to an important and widely accepted standard. Originally it was developed in the context of social networks in order to enhance them with further functionality. The specification encompasses the component hosting environment (container) and a set of APIs that can be used by a Widget. It supports different standard technologies in the Web, such as Oauth for authentication.

For both formats Open Source Widget containers available. Apache Shindig[10] hosts OpenSocial Widgets, whereas Apache Wookie[11] is capable of dealing with W3C Widgets. In order to provide the FInest web-based portal, these containers are not sufficient however. In addition to the mere hosting, a portal has to integrate the different web applications provided by the Widgets in one big application. Technically speaking this means the integration of the DOM elements from the Widgets in the overall DOM model of the portal. With Apache Rave[12] an Open Source implementation of such software is available. It supports both Widget formats and furthermore, provides features like user management or APIs to enable the communication between widgets. An alternative to Apache Rave can be found in WireCloud, which is the Open Source implementation of the FIWARE Composition Editor and Execution Engine. In contrast to Rave, it provides sophisticated support to create Mash-ups based on Widgets. A more detailed discussion of WireCloud can be found in Section 4.1.4.

4.1.3. Interim Technical Architecture

In this section we provide the interim technical architecture of the FInest Front-End. We use the same methodology as presented in Section 3.1 and give a graphical overview about the architecture in Figure 8. In the remainder of this section we will describe the different components of the Front-End in detail.

[8] http://www.w3.org/TR/widgets/
[9] http://docs.opensocial.org/display/OS/Home
[10] http://shindig.apache.org/
[11] http://incubator.apache.org/wookie/
[12] http://rave.apache.org/

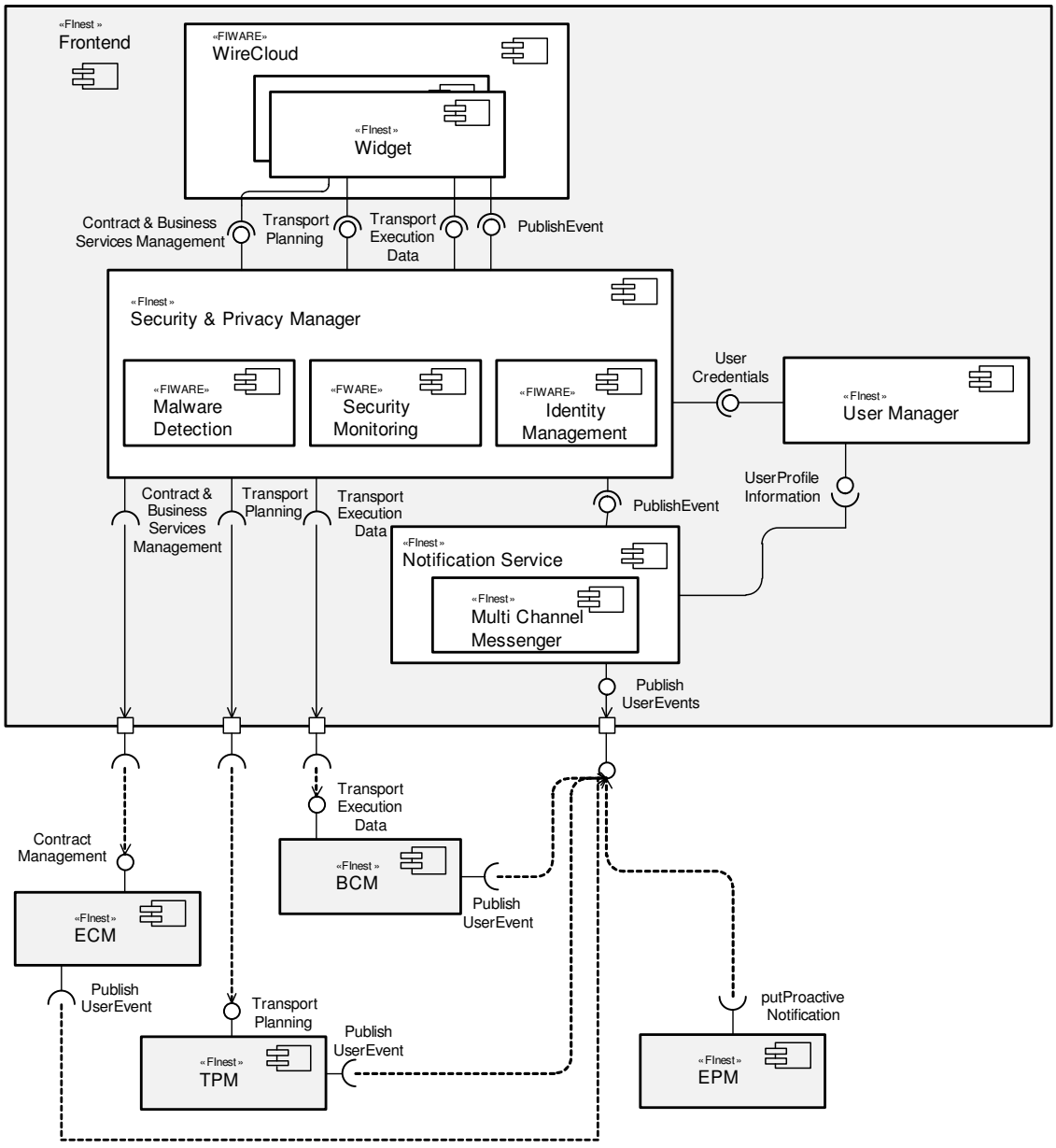

Figure 8 - Interim Technical Specification of the Front-End

4.1.3.1. WireCloud

As we present in Section 4.1.1, the FInest Front-End shall provide a customizable User Interface (UI) for end users. The customization shall be based on Widgets which are provided by the FInest Platform and can be supplemented by external providers so that end users can assemble their UI based on these. The WireCloud Mash-up Platform, which is represented by the WireCloud component in Figure 6, provides the necessary functionalities for this scenario. The platform contains Widgets which use the Web service-based of the Core Modules in order to visualize information gathered by the service calls. For this reason, the Widget components are connected to the Security Manager, which provides the necessary access to the business functionalities. The actual UI is provided by the Widgets and WireCloud provides the portal to select and arrange these.

We mentioned WireCloud directly in the diagram as promising technology for the envisioned UI concept. However, as we stated in Section 4.1.2 there are alternative portals, which implement a similar approach.

4.1.3.2. Security Manager

The Security Manager implements the Security, Privacy & Trust Framework of the FInest Platform. It acts as proxy between the Web service interfaces of the Core Modules and the end user's UI. By this it is able to check security constraints for every client request and ensure that only allowed users have access to the protected information. The Security Manager is taking care of authentication, authentication and data security issues and for this it uses the Security GEs provided by FIWARE. The selection and integration of the GEs is done by the Security, Privacy & Trust Framework. The access to the User Manager enables the Security Manager to get access to user information in order to determine specific access rights.

If a Core module has to present data to the end user or requires an interaction with him or her, it has to provide specific service-based interface which allows the gathering of the relevant data by the UI. In order to implement the necessary security mechanisms for the access to the provided services, the Security Manager has to act as proxy and mimic the provided interfaces of the Core Modules and expose them to be used by the UI.

4.1.3.3. User Manager

The User Manager holds all relevant information about the users of the FInest platform. This encompasses profile information - such as name, organisation/company, business partners or notification setting – but also all security relevant data – such as access rights and login credentials.

4.1.3.4. Notification Service

The Notification Service is a component to enable multi-channel message based on user defined preferences and provided this functionality to the underlying Core Modules. The Notification Service uses so-called Notification Rules started within the user data managed by the User Manager in order to determine how a user wants to be informed about a certain event. For example, an end user can define that he/she want to be informed immediately via SMS if urgent event occur, such as a deviation within a transport process which would require a re-planning. The Notification Service abstracts from the actual way of message delivery and enables an agnostic event distribution with regard to the way of delivery (no difference between send an event via email or SMS). The Core Modules are just concerned with the publishing event of a certain type to the Notification Service and the Notification Service is taking care of the actual delivery based on user preferences stored in the User Manager.

Originally (*cf.* Deliverable D5.3) a similar component was introduced by the BCM, however, the further platform development has shown that also other Core Modules have applications for such a facility. For this reason the Notification Service was introduces to be usable by all Core Modules.

4.1.3.5. Module Interfaces

In the final section we give a description of the interfaces used within the Front-End technical architecture. We use the same table schema as in Section 3.4, except the last column for denoting the realized conceptual interactions, because the Front-End architecture was never described on a conceptual level. See the description of the interfaces within Table 2.

Table 2 – Front-End Interfaces

Interface Name	Summary	Provided by
Contract & Business Service Management	Proxy interface to security and privacy before the corresponding functionality of the ECM is called	Security & Privacy Manager
Transport Planning	Proxy interface to security and privacy before the corresponding functionality of the TPM is called	Security & Privacy Manager
Transport Execution Data	Proxy interface to security and privacy before the corresponding functionality of the BCM is called	Security & Privacy Manager
Publish User Event	Proxy interface to send a user event to the an end-user's client	Security & Privacy Manager
User Credentials	Provides credential data - such as user name, passwords or access rights – to the Security & Privacy manager in order to enable it to check the validity of a client request	User Manager
User Profile Information	Used by the Notification Service in order to determine how an occurred event from the Core Modules or other FInest component shall be sent to the end-user. Contained information determines for an event type the corresponding delivery channel (SMS in urgent cases) for example.	User Manager
Publish Event	Interface to publish a specific event for a specific user	Notification Service
Publish User Events	Interface for the Core Modules and other FInest components to publish events for users; in contrast to the interface 'Publish Event' this event do not have to be specific to a user or channel – the Notification Service will determine this information	Notification Service
Contract & Business Service Management	Interface to access the data about contract and business services contained in the ECM	ECM
Transport Planning	Interface to access transport planning data contained within the TPM	TPM
Transport Execution Data	Interface to access and modify all data with regard to the transport execution contained within the BCM	BCM

4.1.4. Usability Analysis of FIWARE Generic Enablers

As we have shown in Figure 8, the Front-End makes use of different FIWARE Generic Enabler (marked with the <<FIWARE>> stereotype):

- Malware Detection
- Security Monitoring
- Identity Management
- WireCloud

Because the first three handle security issues and such matters are discussed in Section 4.3, we will focus on the WireCloud, which can be used as Widget container and mash-up platform.

4.1.4.1. WireCloud Introduction

WireCloud is the reference implementation of the Composition Editor and the Composition Execution Engine Generic Enablers. It is a tool to create web mashups, which are applications that are integration heterogeneous data, existing application logic and UI components from the Web to a coherent, new application. Thus, it allows the reuse of functionality, scattered over the Web, in order to achieve rapid and cheap development. Although, a specialized tool as WireCloud is not directly necessary to develop Web mashups (manual development), it eases the creation and also can address with little or no programming skills.

WireCloud mashups are based on Widgets, which consist of a combination of HTML (UI elements), JavaScript (application logic) and a Gadget Description Language (GDL) document for the metadata of the widgets. In order to make Widgets available to other users WireCloud allows sharing and trading over an integrated marketplace. This is in direct accordance to the vision of the FInest Front-End, where users shall be able to provide their own interface based on previously developed Widgets. Developers of the FInest platform or from a third party will add content to the marketplace and the user will select (purchase) a set of Widgets in accordance to their very own needs. In addition to this, users and developers can develop mashups by combining the features of available Widgets in a new way and achieve a new application (which also can be published on the marketplace). WireCloud allows this in a graphical way with the provided ‘Wiring-‘ and ‘Piping-Editor’, so that now programming skills are necessary. This is a very easy way for users to customize their user interface and therewith, can be used to enable the vision of a customizable Front-End for FInest.

4.1.4.2. Analysis for WireCloud in FIWARE Testbed

In order to analyse the WireCloud’s suitability for our demands, we went first through all available documentation documents on the official Web site as well as the FIWARE Wiki. After the release of WireCloud in the FIWARE Testbed (mid of August 2012) we were able to conduct a more practical investigation on a running test instance of the framework. Unfortunately, some documentation was still missing, so that we were unable to create and deploy own Widgets. This is considered to be as one very crucial point for FInest, but we do not expect major issues during the process. All in all we discovered the following pros and cons:

Pros

- Comprehensive framework; does almost entirely fit the requirement of the FInest Front-End to provide a customizable UI
- Open source

Cons

- Use of proprietary metadata languages for Widgets (GDL, MDL); hampers reuse of Widgets in other containers (e.g. Apache Shindig); however: support for OpenSocial might be available in third FIWARE release
- Styling and design (templating) options for the created applications are limited so far; hard to achieve a cooperate design if you create a UI.

4.2. System & Data Integration

As FInest is primarily providing a novel infrastructure for allowing efficient and automated interactions between business partners to overcome current limitations in inter-organizational collaborations, such as manual efforts with fax or email and proprietary, peer-to-peer and costly IT integration setups, there is still a need to be able to interact, access and communicate with the existing IT landscape, such as companies' ERP systems, booking systems and domain specific systems such as port and cargo management systems. Furthermore, a major capability of FInest, the online tracking and updating of the execution of a transport plan, requires continuous access to and notifications by various external systems and mainly IoT.

For this, FInest is to provide a technology framework to allow service and application developers to implement concrete integrations for their services to external services, systems and the IoT.

From the study conducted in the previous deliverable D3.2 [3] it was concluded that web services and REST API support would provide sufficient support when including support for EDIFACT and EDI transport. These were the factors which guided the design considerations for this building block of the FInest architecture as well as the understanding that when developing a service or application in FInest, any other building block or specific module may need such an external integration.

4.2.1. Design Considerations

When designing this building block, the basis was for utilizing existing and state-of-the-art approaches and tools and build upon them. Enterprise service bus [4] concept and realization of it by many vendors provides a great mechanism to decouple data and transport considerations when developing services and applications. Adapters for transports and mappers for data structures may be developed and deployed quite easily and systematically without requiring either the external system to be changed or the application being developed to be aware of the specific and sometimes obsolete means for interacting or supporting legacy data structures.

There have also been advances in providing such could-based connectivity support and fortunately FIWARE includes such a generic enabler called Mediator [5].

Another consideration in the design was for various means of inter-connectivity such as request-response, message-driven, publication and subscription, etc. In some cases, multiple modules and services may need to receive and therefore subscribe to the same information or notification, at other cases a service may need to call on a specific external service and query for specific information at a specific point in time. Again the enterprise service bus concept and its realizations support multi-method of interactions.

Finally and as a result of the conclusion from the previous deliverable D3.2, the design considerations included support for specific standards and protocols for the transport and logistics domain, i.e. EDI and EDIFACT as well as connectivity to the Internet of Things. And as such the solution includes, in addition to an ESB, specific adapters and mappers to support EDI as well as a one point interface for all that is IoT based on the FIWARE specifications.

4.2.2. Interim Technical Specification

Following the considerations in the previous section, Figure 9 depicts the technical specification supporting the design.

First and foremost, the design allows for the retrieval of data from external business systems but also allows for providing data from FInest to these systems. The means of interactions may vary, for example, the retrieval may be via posting a query onto the external system or by subscribing to a topic that the external system publishes to. These means are encapsulated by the *retrieveData()* and *provideData()* interactions between the External Business System blocks and the Mediation GE component.

As the core modules and the expected services and applications to be developed in FInest shall support REST interfaces, and as there is need to support domain specific interactions, the design includes specific mediations depicted as EDI2REST and REST2EDI planned to be supported by the Mediation GE. This is depicted in a dotted line as not all integration may require these mediations and therefore may include direct connection between the ESB and the external systems.

Finally, the ability to capture information from devices and sensors is provided via the Pub\Sub GE [6] from FIWARE which is designed for capturing notifications by ‘Things’ as publications and providing them to targets of interest via subscriptions.

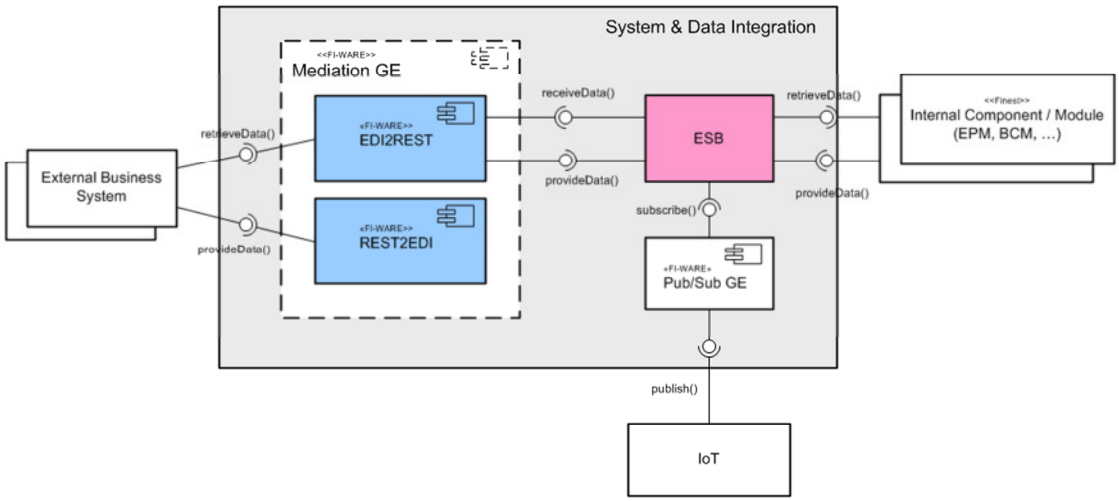

Figure 9 - Interim technical specification of System & Data Integration

4.2.3. Usability Analysis of FIWARE Generic Enablers

As mentioned in the previous sections, two main FIWARE generic enablers were identified as well as the identification of the ESB concept to be of value.

As for the ESB concept and for the integration of external systems with FInest core modules and to be developed services and apps on top, there is a dependency on the middleware that is to support the management and connectivity within FInest. FIWARE has included in its first open call a topic on "Middleware for efficient and QoS/Security-aware invocation of services and exchange of messages" [7] which seems to be addressing the requirements discussed above; however, more details on the approach and specification of the Generic Enablers is expected only after October 2012 and therefore will be addressed in deliverable D3.4 on the Phase 2 implementation plan.

4.2.3.1. Mediator GE

FIWARE's Mediator GE is built using WSO2 Enterprise Service Bus. Mediation tasks such as EDI2REST can be developed, deployed and managed via the generic enabler.

Currently, the specification include Management APIs that describe how to retrieve for example a list of existing mediation services, however, details on how to develop and deploy new mediation tasks as well as interacting with these tasks (e.g. activating a mediation task when calling on a service or when receiving a notification from a subscribed topic) seems to be missing and a ticket has been issued in FIWARE's appropriate tracker.

4.2.3.2. Pub\Sub GE

The purpose of FIWARE's Pub\Sub GE in regard to the Internet of Things is to exchange contextual information such as that of a sensed object by sensors or a tracked object by devices. The specification of the generic enabler includes RESTful APIs description based on OMA NGSI 10 standard for performing for example a subscription on a context of an entity, querying for context information and updating context. It is not clear however how the entire sequencing

needs to be performed against this generic enabler for publishing contextual information and for subscribing to it and therefore the requirements from the ESB to be used on how to interact with this GE (a ticket has been issued in FIWARE's appropriate tracker). Fortunately though, FIWARE is providing means and integrations between the IoT generic enablers and the Pub\Sub GE therefore for FInest it is less important to understand the publication sequence of contextual information and the integration with the IoT.

4.3. Security, Privacy & Trust Framework

In this section we want to discuss the Security, Privacy & Trust framework of the FInest platform. The platform shall be 'secure & reliable by design' – meaning that the platform is designed to be secure by the ground up. Attacks on the platform are taken for granted and the design minimizes the impact of potential security issues. As a consequence, the platform always distrusts external data, only allows user to operate with the minimum of necessary access rights and implements secure message as well as storage facilities. All efforts towards the introduction of a security framework are based on these principles.

Based on the identification of relevant technologies and the usage of the respective FIWARE results (presented in the preceding deliverable, see D3.2), the following presents our work on an interim technical design for the Security, Privacy, and Trust Management Framework.

4.3.1. Security Technologies and Usage Models

In the following table (see Table 2) we want to describe different usage scenarios for the FInest platform. For each scenario we provide a list of security requirements and present relevant security technologies. Additionally, we denote the origin of each technology – which means, if it is provided by FIWARE or if it is an external software component.

Table 2: Security Scenarios & Required Technologies

Security Scenario	Security Requirement	Security Technologies	Version & Release Information
User Management	Authentication of Users and profiling should be managed	IDM, Privacy from FIWARE	IDM from FIWARE Release1 Privacy from FIWARE Release2
Access Control	User shall see only what he / she allowed to see	Role Base Access from FIWARE	Expecting from FIWARE Release2
Private Data	User will have an option to define own data as public or private	Secure Storage from FIWARE	Expecting from FIWARE Release2
Monitoring & Alerting Security Incidents	Service Provider shall see and follow the Security Incidents and take action	Security Monitoring from FIWARE	Expecting from FIWARE Release1

Secure User Data Sharing	User will have an option to choose sharing his/her own data for other parties	Data Handling from FIWARE	Expecting from FIWARE Relase1
Malware Detection	Uploading binary files to the platform should be scanned	Anti Malware from FIWARE	Expecting from FIWARE Relase2
Social Network Security	Links to the harmful sites should be prevented	Defensio	From Third Party
Port / application awareness	Necessary network ports should be opened others should be blocked	Next Generation Firewall	From Service Provider
Attack Prevention	Network and application attacks should be observed and prevented	Intrusion Prevention System	From Service Provider
Web Attacks Prevention	Web server targeted attacks should be observed and prevented	Web Application Firewall	From Service Provider
Denial of Service Prevention	DOS and DDOS should be observed and prevented	DDOS Prevent Systems	From Service Provider

For Authentication, Authorization, Accounting, User Management these Security Services will be used.

- Identity Management
- Privacy Enabler
- Role Base Access (Second release of FIWARE will provide that enabler)
- Attribute and Data Handling enabler

Security Monitoring collect the logs and events from the necessary modules and events will be produced according to the policies.

Optional Security Services like Anti malware and Secure Storage will be used with the necessary applications or widgets. USDL –EXT language will be used for the integration.

Context Monitoring Security & Compliance will use the necessary optional security enablers to provide security. IDM and its extensions like Privacy and Data Handling will supply more user control mechanisms and control. From the SP (Service Providers), we are expecting security solutions from table 2

4.3.2. Interim Technical Specification

After presenting usage scenarios of the FInest Platform, show how we have to address security aspects within it and denote the relevant technologies, we now present a technical specification of the Security, Privacy & Trust Framework. Figure 8 show an overview about it and in the remainder of this section we want to describe it more in detail.

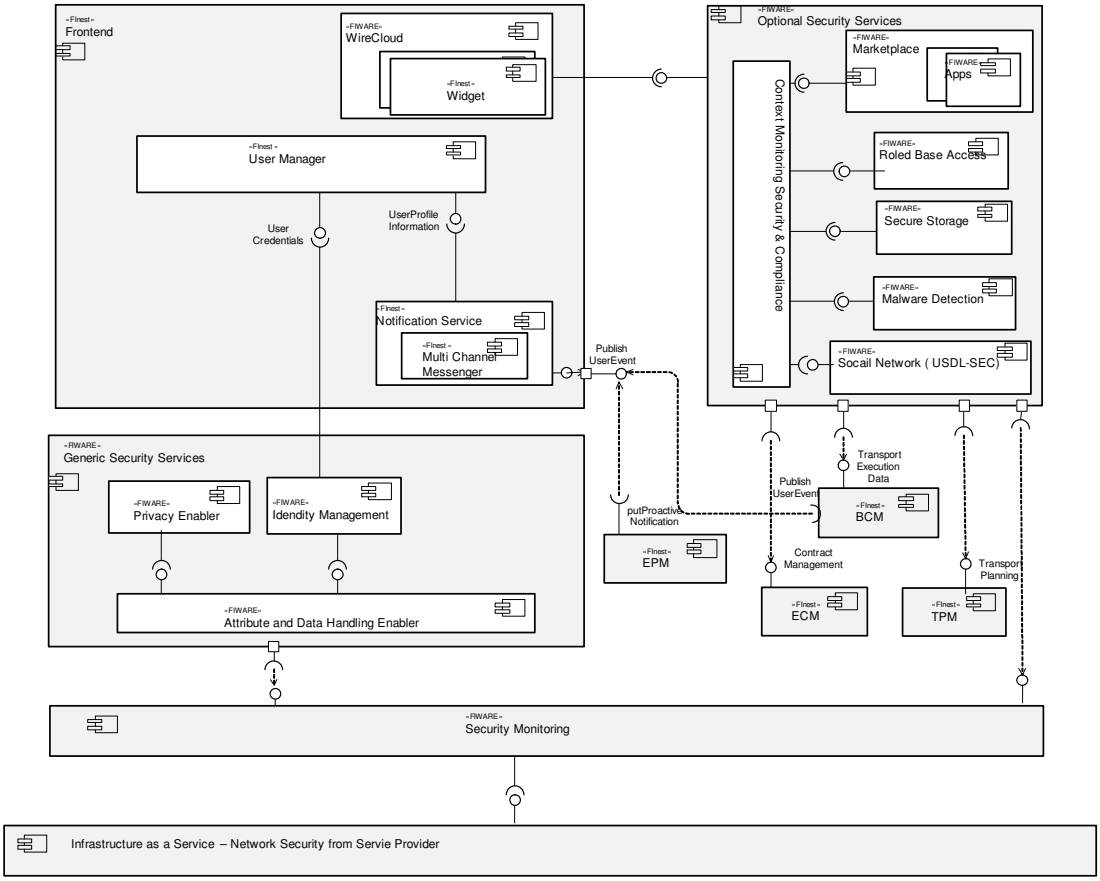

Figure 10 - Interim Technical Design of the FInest Security, Privacy & Trust Framework

IDM GE from FI-WARE (from Generic Security Services) will be main authentication and user management mechanism for the FInest platform. Front End will use IDM and other complementary solutions like Privacy GE, Data Handling GE and Role Based Access GE for the user access and its rights. If there is a more control mechanism is needed from Front End widgets than optional Security GEs will be selected by Context Monitoring and suitable GE will be activated. After the security checks has been done, then main FInest services will run on the platform. Context Monitoring also can choose the best security solution from FIWARE marketplace. Security Monitoring will collect all necessary logs from the platform and Service Provider network elements to provide a real time Incident Management and fraud analyzing. Some of the optional Security GEs will be provided by FI-WARE. On the other hand some solutions will need to be integrated with FInest like Social Network Security. For Application Developers, on Software Developer Kit, we think that suitable security services will be chosen and integrated to the applications. There will be list on the security technologies and developer will have an option to integrate them to the applications.

4.3.3. FIWARE Security Generic Enablers

In this section we investigate the Generic Enables provided by the Security chapter of the FIWARE core platform. We discuss the functionality of each GE and describe the potential application within the FInest Platform.

Security, Privacy and Trust in FI-WARE is mainly focusing on delivering tools and techniques to have a secure design. The high-level Reference Architecture sketched in Figure 11 below is formed by four main modules:

- Context-Based Security & Compliance Services
- Generic Security Services
- Optional Generic Security Services (instantiated at runtime, dynamically reconfigured)
- Security monitoring mechanisms

In the following we describe the potential GEs within each main module and discuss their application within the FInest platform.

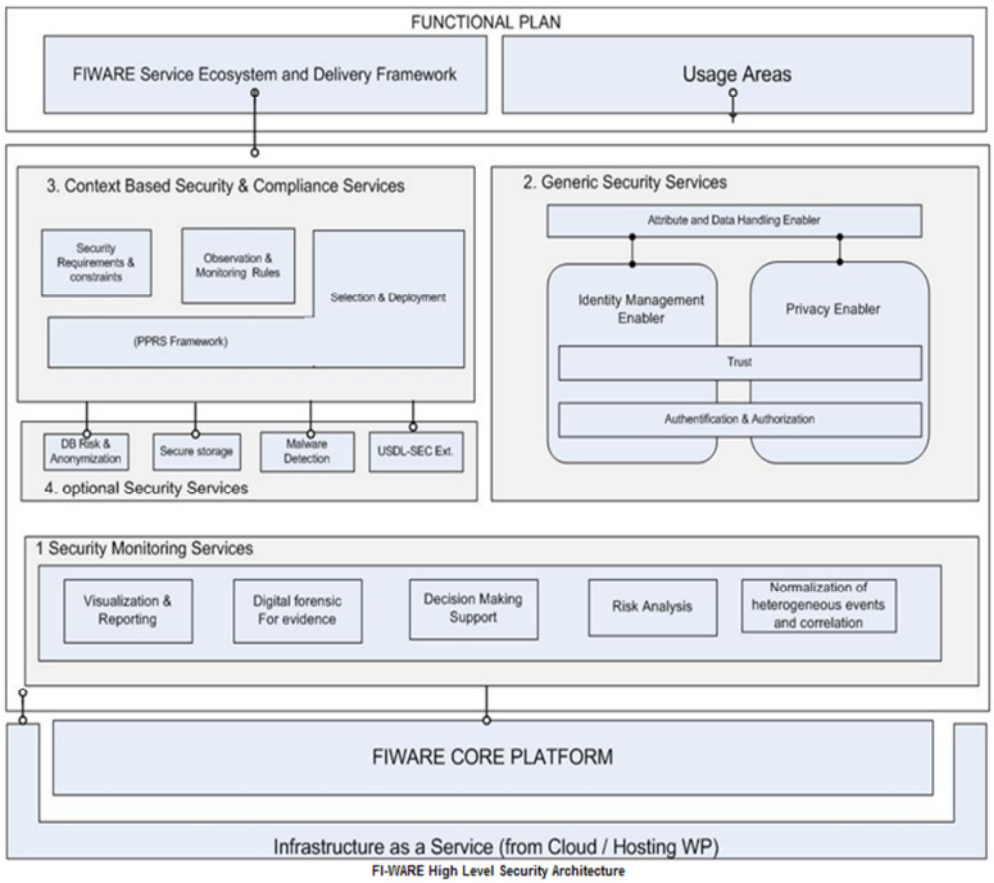

Figure 11 – FI-WARE High Level Security Architecture

4.3.3.1. Context Based Security & Compliance

The GE will accept security request from an end-user application and will select the best Optional Security Enabler to fulfill it by searching the best solution into the Security section from FI-WARE marketplace.

At the same time that the security solution is deployed into the end-user environment the GE also instantiates a runtime monitor with the responsibility of detecting anomalous behavior or non-conformance.

4.3.3.2. Generic Security Services

Identity Management

Identity Management encompasses a number of aspects involved with users' access to networks, services and applications, including secure and private authentication from users to devices, networks and services, Authorization & Trust management, User Profile management, Single Sign-On (SSO) to service domains and Identity Federation towards applications Identity Management is the mandatory part of the FIWARE and also

Privacy Enabler

The Privacy GE does not expose its own external interfaces directly; rather it enhances the functionality provided by the other generic enablers:

- Identity Management GE
- Data Handling GE

It will provide functionality for the Issuer, User and verifier

- Basic credential system
- Selective release of attributes
- Proving predicates over attributes
- Proving predicates over multiple credentials
- Verifiable inspection of attributes

Attribute and Data Handling enabler

The Data Handler GE is a privacy-friendly attribute-based access control system to (sensitive) data. It mainly focuses on revealing certain attributes according to specific prove conditions. It supports integrated data handling (two-sided detailed data handling), that takes into account specific preferences/policies expressed using the PPL language

Data Handling enabler can be used for the find relevant partners or users to find each other by using the privacy policy that is written by PPL. In [8] a usage scenario for the Data Handling GE is explained.

4.3.3.3. Security Monitoring

The Security Monitoring GE is part of the overall Security Management System in FI-WARE and as such is part of each and every FI-WARE instance. The target users are: FI-WARE Instance Providers and FI-WARE Application/Service Providers.

Security monitoring is the first step towards understanding the real security state of a future internet environment and, hence, towards realizing the execution of services with desired security behavior and detection of potential attacks or non-authorized usage. Figure 12 shows an overview about the architecture of FIWARE's Security Monitoring GE, taken from the FIWARE technical specifications.[13]

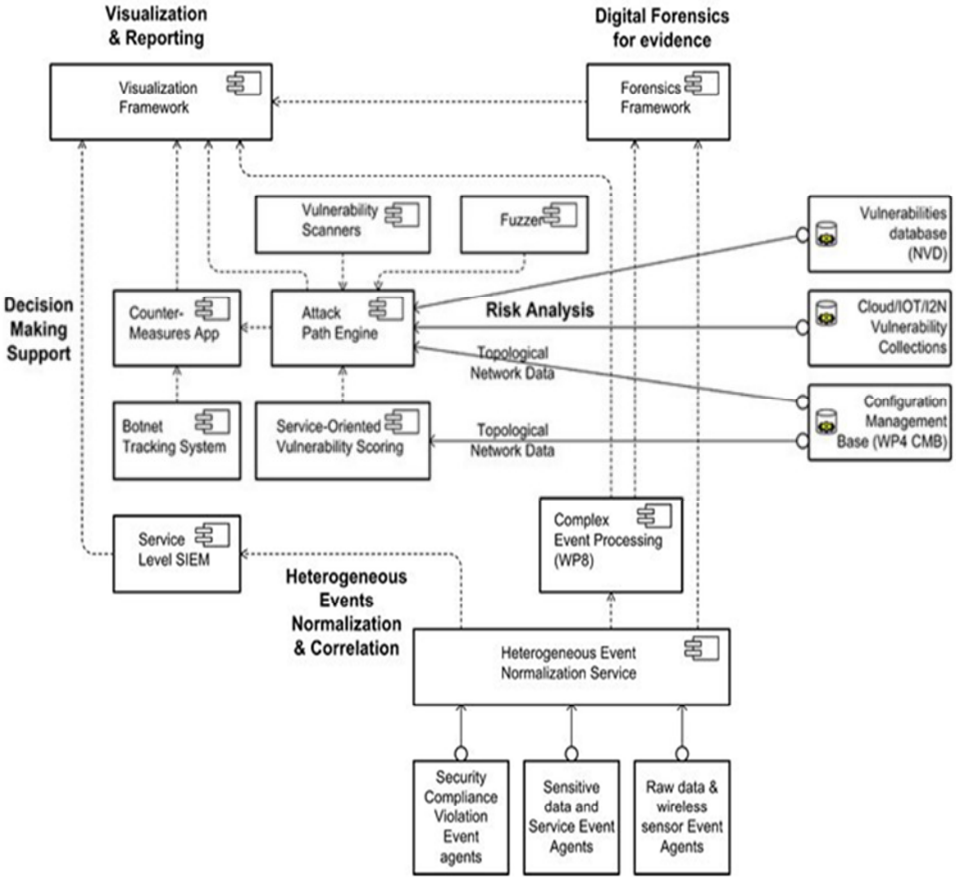

Figure 12 - Architecture of FIWARE Security Monitoring GE

4.3.3.4. Optional Security Generic Enablers

Optional Security Generic Enablers will be used by some part of the FInest platform, i.e. malware scanning before uploading the files to the system, to supply the security. Context based Security GE will also check the suitable optional Security GE to provide to the end users applications. In FInest platform, when needed a new security solution, Optional Security enablers will be designed and it will be placed on FIWARE market place to be used by Finest or

[13] All technical specifications are available on the public Wiki of the FIWARE project: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Architecture_and_Open_Specifications.

any other projects. The available security service features as well as the rules to be fulfilled and the interfaces specification are described by using USDL-SEC language.

Anti-Malware - Morpheus

A malware is a program designed to have an unwanted behavior seen from the legitimate user's side. It may be used to disrupt services, organize valuable/hidden data leaks, or give access to some non-authorized security levels on a system. Morpheus is a detector of malware which takes into account some parts of the structure of programs. Given a database of malware and some input binary code, "Morpheus" verifies that the binary code is not infected by one of the malware of the database

Morpheus reads input binary files and extract from them signatures. Signatures are composed of sites that are abstract descriptions of the behavior of the input program. Binary files are only compared according to their signatures through their sites. The signatures depend on some theoretical options such as dynamic/static analysis, or some security thresholds. The system provides currently only the static/dynamic choice. Morpheus takes into account a white list database that is a list of known and safe signatures. Usually, it contains basic operating system services.

Answers are given either as SANE/INFECTED or by a distance matrix to malware database. The second option offers a finer analysis of suspicious cases. The basic design principles are:

1. Direct Submission through a browser

   Once authenticated, a normal user can submit a binary file by filling a form (local binary path, scan or distance action, mode between static and dynamic). The result is directly displayed in the browser.

2. Web Service client application

   - Client application submits a binary file through scan or distance web service and waits until morphs-engine returns the result (a Boolean or a distance vector).
   - Local binary file is transmitted to server via MTOM (Message Transmission Optimization Mechanism)

In Finest platform, when a binary file will be uploaded to the system, by using the anti-malware solution, it can be prevented to upload the harmful files to the platform

Secure Storage

The Secure Storage Service (*cf.* Figure 13) provides a storage for labelled (i.e. XML DSIG protected) data. It comes with an application-level filter which authorizes read access in function of the identity of the authenticated requester (for example, a service provider) and in function of the sensitivity of the data.

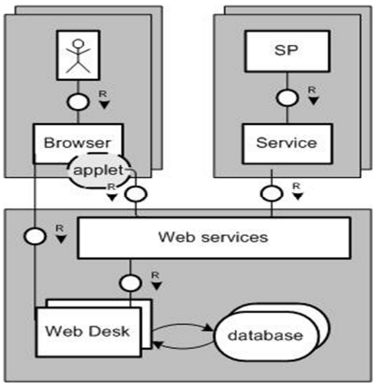

Figure 13 - Secure Storage GE

In FInest, some important information like, conusmers contracts information, can be labelled as private hence just consumer can reach the data.

Role Base Access

Role Base Access control will be released on second version of FIWARE. It will give more access control to the users to reach data and more control mechanisms.

Social Networking

Social Network security is important. Nowadays some harmful links can be published on social network web pages of blogs by using web 2.0. The application should work on page and should be download form app market of FIWARE by users. A nice example of the application is defensio[14], where runs on Facebook or blogs. There is an API extension and it can be integrated.

[14] http://www.defensio.com/what-is-it

5. Conclusions and Outlook

This Deliverable has provided the interim technical specification of the FInest Platform. This encompasses primarily the initial technical architecture with main focus on technical interaction of the main technical building blocks and the Core Modules (complementing the detail technical specifications provided in the M18 deliverables of work packages WP5 – WP8), including the detailed inspection and usability analysis of the relevant Generic Enablers provided by the FIWARE project. In addition, refinements on the overall concept have been presented, particularly on the overall operational model of the FInest Platform and on the identification of additional technical building blocks that appear to be relevant to allow for the commercialization, proper operation, and usability of the FInest Platform. Continuing the domain-driven and iterative methodology applied for the solution design and technical architecture, the parallel work streams concerned with the technical and the conceptual design have closely interacted with the domain experts and aligned the work with the business assessment in WP1 and the refined use case specification and experimentation planning in WP2.

The main achievements and progress from the previous project milestones can be summarized as follows. Firstly, the interim technical specifications present the major step from the conceptual design as presented in M12 towards the technical implementation, including the usage of a substantial set of Generic Enablers developed in the FIWARE project. Secondly, the refined functional specification clarifies the main purpose an operational model of the FInest Platform as whole: the Core Modules provide an initial collection of interoperable components that offer re-usable functionalities upon which domain-specific solutions can be developed in a rapid and cost-efficient manner, where the BCM and EPM enable the real-time provisioning of all information to each stakeholder involved in logistics processes in an event driven manner, the ECM provides a general business functionality for finding and managing business partner relations in open logistics business networks, and the TPM provides a domain-specific capability for transport planning using up-to-date information. Thirdly, with the identification of the additional technical building blocks the refinement of the overall concept can be considered to be completed: the store concept allows for the commercial provisioning and consumption of services and apps within the FInest Platform, the middleware and component management shall allow for a proper and consistent operation of the FInest Platform on the Cloud, and the refined concepts for supporting the personalization, configuration, and customization shall allow for the desired adaptation of the FInest Platform.

With this, the results and progress on the overall solution design and technical architecture can be considered to be achieved as planned, and WP3 continues to serve as the integration point for the various parallel work streams of the project. For the final milestone, it is planned to continue the work, particularly towards the refined technical specifications of the identified main building blocks as the basis for the phase II implementation plans, and the development of conceptual prototypes of the Core Modules exploiting the available Generic Enablers.

6. References

[1] A. Metzger, R. Franklin, and Y. Engel, "Predictive monitoring of heterogeneous service-oriented business networks: The transport and logistics case (best service engineering innovation & quality paper)," in *Proc. of SRII 2012 Global Conference*, 2012.

[2] R. Franklin, Y. Engel, M. Hagaseth, A. Tjora, F. Fournier, R. Fleischhauer, C. Marquezan, A. Metzger, M. Stollberg, and S. Tschapke, "Cloud based collaboration platform for transport & logistics business networks," in *Proc. of 6th International Scientific Symposium on Logistics*, 2012.

[3] R. Fleischhauer, M. Hagaseth, K. Jugel, A. Metzger, A. Rialland, G. Sharon, Ö. Sönmezer, M. Stollberg, V. Verim, and M. Zahlmann, "FInest Deliverable D3.2," 2012.

[4] Wikipedia, "Enterprise Service Bus," 2012. [Online]. Available: http://en.wikipedia.org/wiki/Enterprise_service_bus.

[5] FIWARE, "FIWARE Mediator Generic Enabler," 2012. [Online]. Available: http://fi-ware.cloud.labs.ericsson.net/node/116.

[6] FIWARE, "FIWARE Pub/Sub Generic Enabler," 2012. [Online]. Available: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Data.PubSub.

[7] FIWARE, "FIWARE 1st Open Call," 2012. [Online]. Available: http://www.fi-ware.eu/fi-ware-1st-open-call/.

[8] FIWARE, "Data Handling GE Usage Scenario," 2012. [Online]. Available: http://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Security.Data_Handling_Generic_Enabler.

APPENDIX

The table below provides a comprehensive overview of the current status of the delivery of Generic Enablers by FIWARE (planned and actual) and the planned usage within the FInest project; this information have been elaborated within the Architecture Board; the column at the right indicates how many of the UC-projects are considering the usage of each GE.

Table 3 – FIWARE Generic Enablers - Availability & Usage in FInest

FI-WARE Catalogue: http://catalogue.fi-ware.eu **Last upate (FI-WARE):** 12-Sep-12 Interested UC Projects (should change 'x' for 'E' or 'U', see note 1)				
	Planned/Actual 1st Deployment	Planned/Actual 1st Upgrade	Finest	Total
		data updated on (UC):	19-09-2012	
Cloud Chapter - I2ND (see notes 2 and 3)				
Allocation of VMs	30-Sep-12			4
Allocation of Object Storage	30-Sep-12		U	7
Creation of Cloud Proxies	31-Aug-12		E	4
Data Chapter				
Complex Event Processing (CEP)	23-Aug-12		U	7
Publish/Subscribe Broker	30-Sep-12		U	8
BigData Analysis	10-Aug-12		U	7
Domain Compressed Video Analysis (former Multimedia Analysis)	31-Aug-12			4
Query Broker	15-Sep-12			6
Location Server	10-Aug-12			3
Semantic Application Support	23-Aug-12		**E**	5
Semantic Annotation	15-Sep-12		**E**	5
Apps Chapter				
Service Description Repository	10-Aug-12		U	5
Marketplace	10-Aug-12		**U**	6
Light Semantic Composition Editor	31-Aug-12		E	6
Mashup Factory	23-Aug-12		E	7
Ericsson Composition Editor (ECE)	15-Sep-12		E	7
WireCloud	23-Aug-12		U	5
Mediator	10-Aug-12		U	4
IoT Chapter				
(Backend) Things Management GE	31-Aug-12	30-Sep-12	E	6
(Backend) Device Management GE	(2nd release)			7
(Backend) Advanced Connectivity Management GE	(2nd release)		E	2
(Gateway) Data Handling GE	30-Sep-12		E	5

(Gateway) Advanced Connectivity Management GE	(2nd release)			1
(Gateway) Protocol Adapter GE	30-Sep-12			1
(Gateway) Device Management GE	15-Sep-12			5
Security Chapter				
Security Monitoring GE	30-Sep-12		U	7
Identity Management	31-Aug-12		U	7
Data Handling	23-Aug-12		U	5
DB Anonymizer	10-Aug-12		E	4
Context-based Security & Compliance	????		E	5
Secure Storage	30-Sep-12		U	3
Total	31	1	23	158

1 - UC Projects should replace 'x' (derived from former survery) by:
- cell meaning you have already taken the GE into your Demo PoC:
- cell meaning you have already taken the GE into consideration in your design:
- cell meaning you plan to experiment with it and consider it based on results:

D
U
E

2 - Despite there are multiple GEs, what matters is what are you planning to use the FI-WARE Cloud for

3 - The Cloud Proxy GE is shared between the Cloud and I2ND chapter, therefore merged here