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

D3.1

Technological Requirements and State-of-the-art Analysis

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	UDE	
Editor(s)	Andreas Metzger	UDE
	Michael Stollberg	SAP
Contributors(s)	J. Rod Franklin	KN
	René Fleischhauer	SAP
	Stephan Heyne	SAP
	Clarissa Marquezan	UDE
	Yagil Engel	IBM
	Guy Sharon	IBM
	Volkan Verim	KOC
	Kay Fjortoft	MRTK
	Lone S. Ramstad	MRTK

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 285598

Reviewer(s)	Michael Zahlmann	KN
	Matthias Winkler	SAP
	Guy Sharon	IBM
Dissemination Level	PU	
Contractual Delivery Date	30.09.2011	
Actual Delivery Date	30.09.2011	
Version	1.0 – final	

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 285598

Abstract

This report presents the first Deliverable of work package WP3 "Solution Design and Technical Architecture". It describes an initial version of the Technical Architecture of the FInest Platform (cf. Task T3.2 as defined in the DoW) along with the identification of technical requirements and the examination of candidates for the technical realization (cf. Task T3.1) including the examination of the FI-WARE product vision and the definition of initial requests on Generic Enablers (cf. Task 3.3). In addition, the report defines the procedures and techniques employed for aligning the R&D work in the technical work packages of the project.

Work package WP3 is concerned with the overall design and architectural specification of the collaboration & integration platform for transport and logistics business networks (the FInest Platform). The ultimate aim of the work package is to elaborate on a detailed specification of the overall system architecture that satisfies the business requirements determined in WP1 and WP2 for it to be developed on top of the Future Internet Core Platform in development by the FI-WARE project. In order to achieve this, the scope and main tasks of WP3 are: (1) the functional design and technical architecture specification of the overall FInest Platform with respect to the business requirements in the transport and logistics domain (2) the coordination of the alignment and integration of the core functional modules of the FInest Platform which are developed in work packages WP5 – WP8, and (3) the design of general technical components of the FInest platform relevant for all core modules, including the identification of technical requirements with respect to the desired functionalities determined by the domain experts as well as the identification of Generic Enablers that are to be requested from the FI-WARE project for the technical realization.

Following the iterative R&D methodology deployed within the FInest project, the present deliverable covers each of the aforementioned aspects, providing the initial results that shall be refined and further developed throughout the subsequent milestones of the project. In particular, the report includes

- *the approach, procedures, and modelling techniques employed throughout the FInest project, therewith defining the R&D methodology for the alignment of the technical work packages (Section 2),*
- *the initial conceptual design of the FInest Platform, including the description of the platform's high-level technical architecture and the capabilities and technical interfaces of the platform's core functional modules (Section 3),*
- *the identification of relevant technology areas for the general technical building blocks of the FInest Platform, including the definition of technical requirements and a preliminary investigation of potential candidates for the technical realization (Section 4),*
- *the initial identification of Generic Enablers for the overall FInest Platform (Section 5); the actual feature requests to FI-WARE following the methodology and guidelines as defined in the FI PPP Architecture Board are provided in a separate document.*

Document History

Version	Date	Comments
V0.1	29-07-2011	First draft, AM
V0.2	08-08-2011	Updated after Dresden meeting, AM
V0.3	12-08-2011	Draft for Section 3 & update of Section 1 based on FI PPP paper; Template for Generic Enabler Requests described
V0.4	19-08-2011	Revised Sec 4 with example + minor updates (SAP)
V0.5	23-08-2011	Updated with contribution to Section 4 and some recommendations for changes, AM
V0.6	26-08-2011	Finalized Section 4; AM.
V0.7	02-09-2011	Integrated contributions from SAP, IBM, UDE
V0.8	15-09-2011	Revised complete document, HLA update, integration of partner inputs from KOC, IBM, MRTK, UDE, SAP; MS.
V0.9	16-09-2011	Submission for internal review
V1.0	27-09-2011	Integrated reviewers' comments; ready for submission to EC

Table of Contents

List of Tables

List of Figures

Acronyms

Acronym	Explanation
ASP	Application Service Provisioning (traditional model of software delivery)
B2B	Business to Business
B2B	Business-2-Business
Backlog	Term from SCRUM / agile methodologies: the collection of all themes, epics, and user stories that shall be addressed within a R&D project
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)
ECM	E-Contracting Module
Epic	Type of Backlog Entry: general objective that addresses a particular product and groups several user stories
EPM	Event Processing Module
ETA	Estimated Time of Arrival
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
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
ICT	Information and Communication Technologies
Interface	Technical interface for (automated) information exchange between technical components
INVEST (Acronym)	Acronym for the 6 properties of well-defined 'User Stories' that shall be defined within the FI-WARE Backlog: Independent, Negotiable, Valuable, Estimatable, Small, Testable
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)

OMG	Object Management Group, technology standardization body, see ww.omg.org
PaaS	Platform as a Service
PCS	Port Community System
REST Service	A Web Service accessible as a Web Resource, applying to the principles of Representational state transfer (REST)
PKI	Public Key Infrastructure (security technology)
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
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
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).
Theme	Type of Backlog Entry: a top-level objective that may span several projects and products
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
User Story	Type of Backlog Entry: specific objective for a product, addressing a particular feature request from a user perspective, defined in form of INVEST criteria
VPN	Virtual Private Network (secure & reliable connections for information transfer over the Web)
WSDL	Web Service Description Language (W3C Recommendation)

1. Introduction

The efficiency of international transport and logistics networks is a crucial factor for sustainable growth in global trade. Inefficient operation creates obstacles for the global trade by causing delays in deliveries and increasing costs all over the supply chain management process. Since transport and logistics activities account for 10% to 20% of a country's Gross Domestic Product, highly efficient management tools can dramatically improve competitiveness. In addition, environmental impacts resulting from the operation of transport and logistics activities are significant. Any improvement in efficiency within the logistics network positively contributes to the reduction of the global environmental impacts of transport and logistics domain.

While the transport and logistics industry has made great strides in attempting to improve its efficiency, limitations in technology, transport infrastructure and regulatory regime incompatibilities have created significant barriers to future improvements. Overcoming these barriers requires new information and communications technologies that allow organizations to rapidly assemble collaborative logistics networks that can efficiently and effectively execute international trading activities. The Future Internet, with its promise of ubiquitous operation and information access, provides a promising candidate to overcome the current limitations.

Building on the proposed capabilities of the Future Internet being developed under the European Union's Future Internet Public Private Partnership program (FI PPP), the FInest Use Case project is designing a collaboration and integration platform for the transport and logistics industry. The *FInest Platform* leverages generic capabilities, so called generic enablers, provided by the Future Internet and implements a domain-specific configurable set of services for the transport and logistics domain.

This report presents the first Deliverable of work package WP3 "Solution Design and Technical Architecture" that is concerned with the iterative design and specification of the technical architecture of the FInest Platform. In order to provide a self-contained overview on the scope and position of WP3 within the FInest Project, the following first explains the aims of WP3 and its relationship with the other work packages, and then, with respect to this, outlines the structure of the remainder of the deliverable.

1.1. Aims and Scope of WP3

Work package WP3 is concerned with the overall design and architectural specification of the FInest platform. The ultimate aim of the work package is to elaborate on a detailed specification of the overall system architecture that allows satisfying the business requirements determined in WP1 and WP2 for it to be developed on top of the Future Internet Core Platform in development by the FI-WARE project. In order to achieve this, the scope and main tasks of WP3 are: (1) the functional design and technical architecture specification of the overall FInest Platform with respect to the business requirements in the transport and logistics domain, (2) the coordination of the alignment and integration of the core functional modules of the FInest

Platform which are developed in work packages WP5 – WP8, and (3) the design of general technical components of the FInest platform relevant for all core modules, including the identification of functional and technical requirements and the examination of Generic Enablers to be requested from the FI-WARE project for the technical realization.

For this – in accordance the objectives as defined in the FInest DoW – WP3 will derive the technological requirements for addressing the business requirements identified in WP1, investigate the state-of-the-art on relevant technologies, define a conceptual design and technical architecture for the envisioned technological solution, identify the Generic Enablers required from the FI PPP core platform, and specify the necessary domain-specific capabilities that shall be built on top of this. The result will be a detailed specification of the envisioned collaboration and integration platform for transport and logistics along with a detailed implementation plan for the follow-up project in phase 2 of the FI PPP.

In addition, WP3 is responsible for the coordination and alignment between the technical work packages that developed the core modules of the FInest Platform (WP5, WP6, WP7, and WP8). The aim is to ensure that the core modules will 'by design' be able to interact and be combinable in order to adequately support the procedures and challenges arising within the FInest use cases (defined in WP2) as well as similar ones in the transport and logistics domain (cf. business requirements and general domain requirements identified in WP1). For this, WP3 defines the overall architecture of the FInest Platform with respect to the capabilities and technical interfaces of the core modules that are developed in WP5 – WP8, and selects the common technology baseline for realizing the FInest Platform, therewith providing the governance model and procedures for the technical R&D work within the FInest project. Besides, WP3 is closely related to the FI PPP alignment activities in WP9 (esp. for the identification and feature requests for Generic Enablers), and, together with WP5 – WP8, provides the necessary information on employed technologies and technical demands for the identification of suitable experimentation environments that are inspected within WP4.

Please note that after a thorough reassessment of the initial interaction of WPs in FInest, the updated relationships among the work packages have been agreed by the project management board and the consortium members. The business requirements and scenarios will directly be delivered to the technical work packages (WP5 – WP8) without intervention of WP3: each of WP5 – WP8 is responsible for a thorough state-of-the-art analysis, while the role of WP3 is to coordinate the process and to consolidate the outcomes; this presents an additional key activity of WP3 with respect to the FInest DoW.

1.2. Deliverable Overview and Structure

With respect to the revised aims and scope of WP3 as explained above and following the iterative R&D methodology deployed within the FInest project, this Deliverable covers each of the mentioned aspects, providing the initial results that shall be refined and further developed throughout the subsequent milestones of the project. In particular, the report includes:

- Section 2 defines the procedures and modelling techniques employed throughout the FInest project, therewith defining the R&D methodology for the alignment of the technical work packages (cf. technical governance as explained above);
- Section 3 presents the initial conceptual design of the FInest Platform, including a high-level technical architecture and the description of the functional core modules subsystems with respect to their respective capabilities and technical interfaces (cf. Task T3.2 of the DoW);
- Section 4 identifies the relevant technology areas for the general technical building blocks of the FInest Platform, including the definition of technical requirements and a preliminary investigation of potential candidates for the technical realization (cf. Task T3.1 of the DoW);
- Section 5 presents the initial identification of Generic Enablers for the overall FInest Platform that are requested from the FI-WARE project; the detailed feature requests to FI-WARE following the methodology and guidelines defined in the FI PPP Architecture Board are provided in a separate document;
- Section 6 concludes the report and provides an outlook to the next milestone;
- The Appendix (Section 8) provides additional material on
    - The examination of potential candidate technologies for Graphical User Interfaces for Future Internet Applications (related to Section 4)
    - The template and instructions for Generic Enabler Feature Requests as defined by the FI PPP Architecture Board (related to Section 5).

2. Approach for Technical Coordination

This section describes the alignment and coordination of the technical design and development activities between WP3 and WP5 – WP8 wherein the central components of the FInest Platform (called 'Core Modules', cf. High Level Architecture in Section 3.1.2.2) are defined.

The aim is to ensure that the core modules are interoperable (technical interoperability) and can work together (functional interoperability) to support the transport and logistics processes addressed within FInest, in general for the domain of international freight transport, and – with respect to extended usage of the FInest Platform and technologies – also within other domains where multiple organizations need to collaborate to achieve the business goals.

To ensure this, WP3 defines the overall architecture of the FInest Platform, selects the technological basis for ensuring technical interoperability of the core modules as well as the management and extensibility of the FInest Platform with additional components, defines the methodology and modelling techniques to be within the FInest project, and – most importantly – defines the capabilities (= main features) as well as the interfaces for automated information exchange among the core modules and platform components.

This is the central task of WP3 in order to ensure the interoperability and functional suitability of the FInest Platform with respect to the business requirements determined in WP1 and WP2. The following describes the procedures, methodologies, and tools used for this.

2.1. Alignment and Coordination Process

The following describes the alignment and coordination process of the work packages, with particular focus on those WPs concerned with developing the FInest Core Platform and its core modules. This shall ensure the interoperability of the developed technical components, and the structure presented here has been accepted by the FInest Project Management Board.

2.1.1 Work Package Interactions and Responsibilities

WP3 interacts heavily with other work packages. Important input is coming from WP1 and WP2 while that is used to generate technical governance guidelines for the technical work packages WP5-WP8. Figure 1 below illustrates the relevant interactions and dependencies.

WP1 and WP2 are responsible for collecting general domain (or business) requirements from the domain-partners involved in the project. These are relevant throughout the entire project, especially with respect to the fact that the FInest is – in contrast to most other research projects – not technology-driven but application-domain-driven. That means that a predefined technology is not the basis of the project, rather only the most suitable technologies are used ("technology pull"). As the basis for this, WP1 and WP2 will identify the relevant business requirements as well as concrete use cases that will be addressed by the FInest Platform.

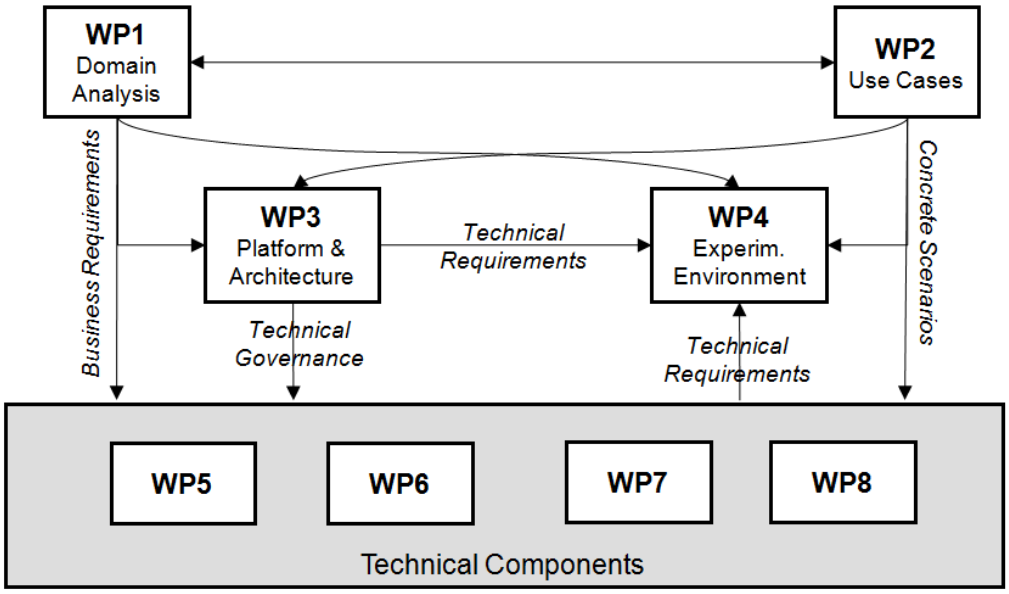

Figure 1: Relationship and Dependencies of FInest Work Packages

2.1.2 WP3 as the Central Technical Work Package

As mentioned above, the responsibilities of WP3 include the development of the high-level architecture of the FInest Platform. Therefore, the capabilities and technical interfaces of the planned modules (e.g. e-contracting component) are defined and described. It is WP3 responsibility to provide guidelines to the definition and description of these modules. WP3 is not solely responsible for providing the specific content. It will collaborate with the technical work packages WP5-8 to align the capabilities and the technical interfaces.

WP3 ensures that the technical work packages will be aligned to the overall plan by using various means of interactions. Those include bi-weekly conference calls where the synchronization with all technical work packages is discussed. At least all technical work package leads attend those conference calls and tasks will be assigned as well as issues with the alignment will be addressed. Further, dedicated face-2-face meetings are organized by WP3 in order to enable focused technical discussions.

The process is an iterative one, which will result in a refined architecture with increasing detail level (also see Section 2.1.3). The overall platform architecture (created by WP3) will serve as a top-down view of the overall solution whereas the Core Module architectures (created by WP5 - WP8) will serve as a bottom-up view. Thereby, the architecture will be continuously reviewed, improved, aligned and enriched with additional details from different perspectives. The process and the details of a single iteration are sketched in Figure 2:

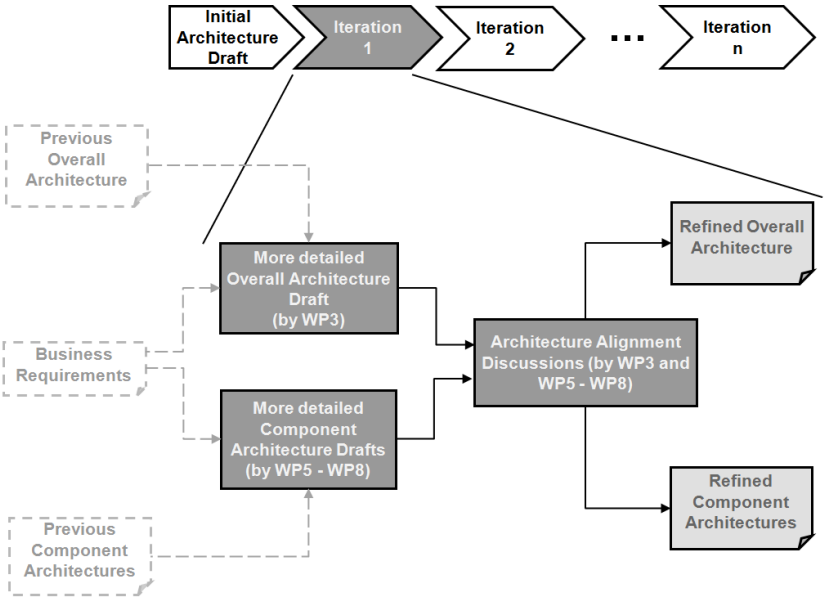

Figure 2. Iterative Process of Architecture Refinement

In addition to overseeing the alignment of capabilities and technical interfaces, WP3 is responsible for selecting a technology basis for common features of the FInest Platform components, thereby fostering the technical interoperability and extendibility of those components. As an example, this includes technologies which are used to integrate legacy systems or to build graphical user interfaces.

Finally, WP3 is responsible for strategic decisions, such as defining the methodology and modelling techniques to be used within the project. The selection of an architecture description language will be described in Section 2.2.

Another strategic task of WP3 is the identification of Generic Enablers that may be used as is from the FI Core Platform, require additions to those provided by the FI Core Platform or that will not be supported by the FI Core Platform but are still crucial for FInest. The identification of those GEs is driven by the technical requirements of the core modules and the overall FInest Platform. The identification process and results are presented in Section 5.

2.1.3 Focus for the Project Milestones

For each milestone as defined in the FInest DoW (M6, M12, M18 and M24), one or more iterations (as described above) will be performed:

- For milestone M6 the special focus lies on the definition of high-level technical component capabilities and an initial high-level architecture of the whole platform extension. Additionally, for each technical component initial ideas are collected for their in-

terfaces. This is complemented by a state-of-the-art analysis and the definition of technical requirements for the platform extension and the components.

- For milestone M12 the conceptual design of the entire domain-specific extension of the FI platform will be addressed and iterated based on the initial results of M6. This includes the iterative detailed definition of component interfaces, especially for those which are relevant for inter-work-package interactions.
- For milestone M18 a well-defined interim technical specification for the entire FI platform extension will be the focus. This includes the interim technical specification of all components.
- For milestone M24 the technical specification shall be finalized and the implementation plan will be in place for the follow-up project in Phase II of the FI PPP.

2.2. Architecture Description Language

The FInest Project Management Board (PMB) and all technical WP Leads have agreed to use Technical Architecture Modeling (TAM)[1] as modeling standard for architecture diagrams in the project. TAM is a UML2.0 profile with additional extensions from Fundamental Modeling Concepts (FMC). For detailed information on TAM please see the website as well as the detailed specification available online[2].

TAM can be used on two different levels, on conceptual level to describe high-level architectures in low detail and on design level where TAM is used to describe software systems from the (more technical) architecture down to the very little details. In contrast TAM can also be used to model non-technical processes and scenarios like business processes. For some more information please see the public TAM website2.

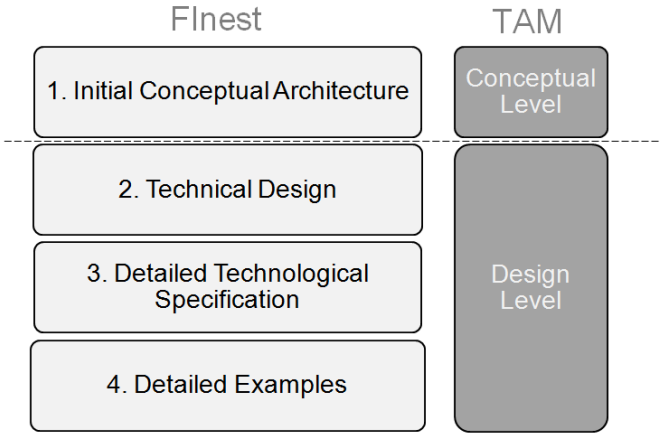

Figure 3: Modelling Levels of FInest and TAM

[1]http://www.fmc-modeling.org/download/fmc-and-tam/SAP-TAM_Standard.pdf
[2]http://www.fmc-modeling.org/fmc-and-tam

In FInest TAM will be used on four different modeling levels as shown in Figure 3. On each level, different TAM diagrams are used as follows:

- For the initial conceptual architecture *block diagrams* are used to design the component and the overall architectures in a conceptual level and without any technical details. *Component diagrams* should only be used to give an outlook to the later design. Additionally to *block diagrams*, state diagrams and *use case diagrams* can be used.
- For the technical design level in FInest the block diagrams will be extended with elements from *component diagrams* to give a more technical view on the previously created architectures. If applicable and necessary in some cases also *sequence diagrams* and *activity diagrams* are allowed.
- For the detailed technological specification mainly package diagrams and *class diagrams* will be used in order to complete the more abstract diagrams of the higher modeling levels.
- For illustrating technical and conceptual details decupled from the architecture and making clear the usage of components and classes detailed examples will also be created in FInest. Those will mainly be created using *sequence diagrams* and in some cases *class diagrams*.

The description of the single diagram types is out of scope of this section. Therefore, please see the TAM Specification for a detailed description. [3]

[3] http://www.fmc-modeling.org/download/fmc-and-tam/SAP-TAM_Standard.pdf

3. Initial Conceptual Architecture

This section presents the initial conceptual architecture of the FInest Platform, which has been derived co-jointly from discussions among the technical work packages with iterative reviews and feedback from the domain experts and with respect to the business requirements from WP1 as well as the use cases elaborated in WP2 of the FInest project (also see Section 2).

The section is structured as follows: Section 3.1 introduces the initial high level architecture of the FInest Platform, along with the central characteristics of the envisioned solution underlying design principles explanations of the main components. Section 3.2 refines this architecture by describing the initial set of services developed for the transport and logistics domain, which correspond to the four core modules that are developed in WP5 – WP8; a more detailed description along with the state-of-the-art analysis for each module is provided in the deliverables of the respective work packages.

3.1. Initial Technical Architecture of the FInest Platform

To introduce the initial technical architecture of the FInest Platform, the following first explicates the basic characteristics of the envisioned platform for optimizing the collaboration and integration throughout business networks within the transport and logistics domain, and derives the basic technical design principles for the FInest Platform. Then, we introduce the high-level architecture that identifies the main components and technical building blocks, along with the definition of the capabilities (= what each component does) and the interfaces (= which information shall be interchanged), and finally outline the suitability and necessity of employing Future Internet technologies for the technical realization.

3.1.1 Characteristics and Design Principles

As stated in the FInest DoW, the overall aim of the FInest project is to develop a Future Internet enabled ICT platform that shall facilitate optimizing the collaboration and integration within international transport and logistics business networks. For this – and under consideration of the business requirements as well as the shortcomings of the ICT infrastructures currently deployed in the domain (cf. Deliverable D1.1) – the following basic characteristics of the envisioned FInest Platform have been identified.

The results of the project shall be …

- … a FI-enabled platform (not a ready-to-use solution) that provides ICT facilities for enabling optimized planning, execution, control, etc. of logistic processes
- … on top of which end-users (= e.g. KN) can quickly develop customized solutions
- … suitable for optimizing the preparation, planning, and execution of transport and logistics for all FInest use cases + every similar use case.

From this, the following basic design principles for the FInest Platform have been derived:

- Technology Pull: 'capabilities' as demanded by domain experts (not 'push' by ICT)
- FInest Platform = extensible collection of FI-enabled components for optimizing logistic processes + technical infrastructure for usage and integration
- Loosely coupled & interoperable components:
    - Each component provides the corresponding capabilities required by all FInest use cases (= foreseen usage) as well as for similar use cases (= unforeseen usage)
    - The components can be selected and aggregated / composed for each individual use case with minimal effort
- Modular & decoupled design for cost-efficient application development & maintenance and extensibility of components
    - Every 'feature' in separate technical component (cost-efficiency)
    - Strong decoupling of UI <-> Business Logic <-> Functional Components
    - Separate middleware management for modules (add / remove / modify)
- Decentralized architecture: the components may be deployed on a cloud environment, and can be accessed by each end-user application over the Web (cloud-based system).

3.1.2 High-Level Architecture

Based on the aforementioned design principles, we have designed the initial conceptual architecture of the overall FInest Platform in form of a high-level architecture that defines the main technical building blocks along with the primary interfaces for information exchange. This has been derived from extensive discussions among the ICT experts with detailed reviews and feedback from the domain experts to ensure the suitability with respect to the business requirements arising in the domain.

The high-level architecture presented here for the M6 milestone is a refinement of the one presented in [18]. It will serve as the basis for further refinement and for the detailed technical design that is planned for the next milestone of the project. Here, we focus on the functional aspects (i.e. identification of main components and their relationships). The selection of specific technologies as well as the detailed technical design will be subject to the next release in the subsequent milestones.

As shown in Figure 4, the initial high-level architecture consist of three layers (Front-End, Core Modules, and Back-End) that encompass the main technical components, which are described in more detail below. In addition, a Middleware Management component is defined that shall

encompass the necessary facilities for supporting the addition, removal, or modification of technical components (Component Registry & Management) as well as the support for deployment in cloud environments and the development of customized user applications on top of the FInest Platform (Cloud Deployment & Management).

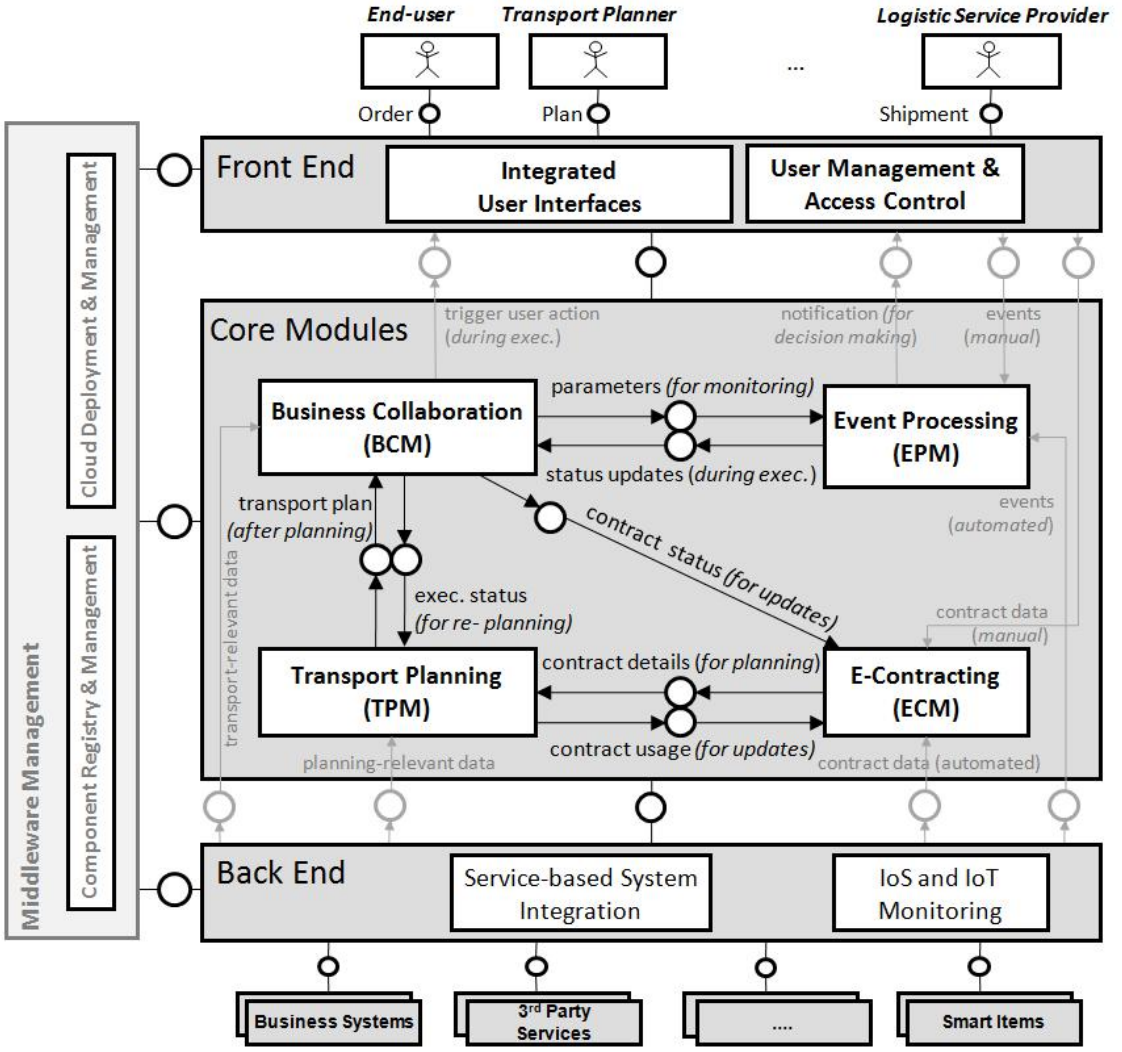

Figure 4: High-Level Architecture of the FInest Platform[4]

3.1.2.1. Front End

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 networks.

The roles that are supported through this front end layer include, but are not limited to, the following: *end-users (customers)* who can issue or trace transport orders; *transport planners* who can develop, monitor and update the overall transport plan and logistics processes for

[4] The notation used throughout this document is TAM (Technical Architecture Modeling), cf. Section 2.2.

solving a customer orders, which may include the individual legs of a shipment, the mode used for that leg of the shipment and the provider of the transport service for that leg of the shipment; *logistics service providers* who can provide offers for transport and logistics services and retrieve demands for these services.

In order to support these roles in using the FInest Platform and its specific features, the Front-End encompasses the following central components

- The Integrated User Interface shall provide a single point of access that integrates the various graphical user interfaces for the Core Modules as well as additional user-tools for supporting the collaboration among stakeholders in transport and logistics business networks; in order to allow for ubiquitous access, the user interfaces shall support access with different channels and devices (web, mobile, desktop, etc.); moreover, the integrated user interface shall be easily customizable for each individual user;

- The User Management & Access Control shall provide the facilities for registering and managing users that are subscribed to the FInest Platform, and ensure the access control to both functional facilities provided by the platform and – more importantly – to business-relevant information (each user shall only have access to the information that (a) he *needs to know* in order to conduct his specific job in a logistics process, and (b) that he *is allowed to know* with respect to confidentiality agreements that are agreed by contract); this requires sophisticated security and privacy management techniques, which shall be employed throughout the whole FInest Platform to ensure access control and information confidentiality.

3.1.2.2. Core Modules

The middle layer of the FInest Platform encompasses the functional core modules that provide the basic functionalities for optimizing the collaboration and integration throughout transport and logistics business networks. These so-called 'Core Modules' shall be provided in form of independent cloud-based services that each provide a set of capabilities that can be combined in order to support the specific demands for optimizing the preparation, planning, and execution of transport and logistics processes in individual use case scenarios. Corresponding to the technical work packages WP5 – WP8, the initial set of core modules that are being developed for the FInest Platform include the following:

- Business Collaboration Module (BCM) – this module keeps all the information that is needed for executing a logistics process where various different business partners are involved in with role-based access control; it therewith allows for the end-2-end visibility of transport and logistics processes, which is one of the key domain requirements

- E-Contracting Module (ECM) – this module provides computer support for service provider selection, contract negotiation and agreement, contract management and

the provision of contract related service requirements to other modules that utilize this information for ensuring the effective and efficient network operation.

- Event Processing Module (EPM) – this module will enable real-time tracing and advanced control for the planning and execution of logistics processes through event-driven monitoring across domains and transportation modality. The module is also responsible for SLA monitoring (based on data from the ECM) and the rule-based analysis of expected and unexpected events, such as e.g. triggering re-planning when needed.

- Transport Planning Module (TPM) – this module provides support for dynamic transport planning and re-planning activities. The main added business value shall be the support for exploiting the relevant and the most recent information for the semi-automated transport planning activities, including real time event data provided through the EPM, contracts between business partners that are managed within the ECM, the current status of a logistics process from the BMC for transport re-planning, as well as from external services or systems (e.g. transport offers from spot marketplaces).

The main capabilities and interfaces as well as the domain-driven relevance for each of the four core modules are explained in more detail below in Section 3.2.

3.1.2.3. Back End

The back end layer of the FInest Platform provides access to, and integration with, legacy systems, third party services and any Internet of Things (IoT) devices that may provide information during the transport lifecycle. This appears to be highly desirable, as various information that is relevant for preparing, planning, and executing logistics processes is available within currently used business systems (e.g. ERP or CRM systems) as well as in 3rd party systems (e.g. global trade services, traffic reporting and weather forecast services, etc.); this information shall be made accessible by the FInest Platform and particularly to the Core Modules, while the existing systems can be continued to be used.

Legacy system integration is facilitated by service-oriented technology, e.g., by exposing features of legacy systems as services, or by offering access to legacy systems via the "Software as a Service" [1, 2] delivery model; the Service-based System Integration component shall provide the basic technical infrastructure for enabling this. In addition, the IoS and IoT Monitoring component shall support the monitoring and analysis of the usage and information exchange of the FInest Platform with the integrated external systems.

3.1.3 Usage of Future Internet Technologies

In order to enable the cost-efficient and easy development of cloud-based user applications on top of the FInest Platform, emerging techniques for the Internet of Services, Internet of Things, and Internet of Content shall be employed. The following enlist the Future Internet technolo-

gies of primary importance for implementing the FInest Platform; most of these are planned to be provided by the Future Internet Core Platform that is developed in the FI-WARE project; while we here depict the primary relevant Future Internet technologies, the detailed initial investigation of relevant and desirable Generic Enablers for realizing the FInest Platform is provided in Section 5 of this deliverable.

- Infrastructure, methodology, and tools for cloud-based platforms and application development, including an infrastructure for deploying the FInest Platform and its components on public or private clouds along with methodology and tool support for developing additional end-user services for individual transport and logistics stakeholders;

- Language and tool support for the Internet of Services (IoS), including a service description language that covers both technical and business requirements along with integrated tool support for the provisioning, management, and consumption of services; this shall be used for realizing the back-end layer of the FInest platform and for managing the interaction of the FInest core modules;

- Access to real-world data from the Internet of Things (IoT), enabling the integration and technical handling of real-world data obtained from sensor networks for real-time monitoring and tracking during the execution phase of transport and logistics processes;

- Facilities for data and event processing, allowing to process huge amounts of data to retrieve insights into relevant scenarios, as well as to analyse real-time event data to quickly determine relevant situations and instantly trigger actions.

- An integrated framework for security and privacy management for the Future Internet, including identity management, authentication and authorization facilities, non-repudiation services and policy management for user profiles as the basis for ensuring the security, privacy and confidentiality of information exchanged between business partners, which is a pre-requisite for employing the FInest Platform in real-world business environments.

3.2. FInest Platform Core Modules

The details for each of the FInest Platform core modules, as introduced above, are further elaborated in the following sections. The focus in these descriptions is on the conceptual architecture and the main capabilities of the core modules; the technical realization and actual usage of Future Internet technologies is subject to future work.

3.2.1 FInest Business Collaboration Module (BCM)

Logistics processes are distributed and involve numerous different stakeholders. Stakeholders may include customers (such as the consignor and the consignee), one or more transport planners, and a set of actual transport providers (carriers or shippers), insurance companies and other legal parties, governance authorities (e.g., customs or border control) as well as other partners. Each one of these parties needs information about the goods being shipped in order to successfully conduct a transport process. However, currently employed ICT systems have been developed for intra-organizational management and do not provide easy access for external partners. Coordination between the different stakeholders, therefore, requires manual intervention in order to share information. The large amount of manual intervention required hampers effective supply chain management and increases the likelihood of errors as well as shipping costs.

The *Business Collaboration Module* (short: BCM) introduces an infrastructure to securely manage end-to-end networks between transport and logistics partners. It integrates information from different sources – such as the *Transport Planning Module* (TPM), the *E-Contracting Module* (ECM) and the *Event Processing Module* (EPM) as well as external legacy systems (e.g. ERP) and user input – and makes it available for the different stakeholders. The main task of the BCM is to provide an overview of the current status of logistics processes and it acts as the central data storage component of the FInest platform. All information relevant to a specific logistics process is kept in a centrally managed storage, with access-control and provides customized views on the data for each involved stakeholder.

To enable this, the BCM uses so-called Collaboration Objects (CO) that implement a data-centric modelling approach [3, 4]. Each CO encapsulates information about a certain aspect of the overall transport and logistics chain (e.g., a certain transportation leg or an involved carrier) and the process fragment associated with this aspect. Hence, a CO consists of two different elements: a data element and a process or lifecycle element. The combination of different COs describes the end-to-end transportation process and establishes a global view of the entire process. In addition, the distribution of information about the various aspects of the transport process over multiple COs enables privacy management due to the fact that only the information that is contained in the particular process aspect which a stakeholder is authorized to see is actually presented to this very stakeholder.

The general functionality of the BCM can be described as follows and is depicted in the conceptual architecture shown in Figure 5 below:

- Create a representation of the end-to-end transport and logistics process – To create a representation of the end-to-end process the BCM initializes the COs by integrating data from existing business (legacy) systems, transport plans from the transport planning system and end-user inputs.

- Store the CO-based process representation – The BCM stores the CO-based representation of the end-to-end process at a logically central, but physically shared, place (e.g. distributed database).
- Provide secure access to process data – Business partners have access to the shared database, but only to those stored COs that their user rights allow. The BCM uses the scattering of information among the COs to ensure security and only discloses the necessary objects to the client.
- Trigger user action– The likelihood of occurring changes during the execution of a logistics process have a relatively high impact. In most cases, different options are possible for reacting to an unexpected situation. The BCM informs users about such circumstances and lets the users decide how to react (potentially imposing changes to the process).
- Provide information about contract status – Special kinds of contracts in the T&L domain target at the amount of shipped goods and not at the conduction of a single process. For this reason, the BCM provides information about the contract status and makes it available for other modules (e.g. ECM), so that it is possible to observe how many transports have already been handled by a single contract.

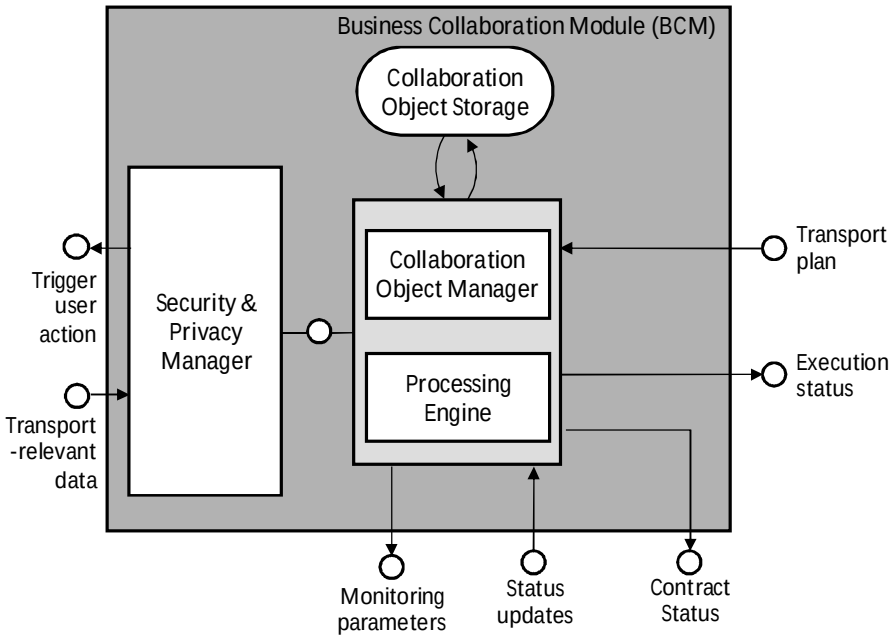

Figure 5: Business Collaboration Module (BCM) conceptual architecture

3.2.2 FInest E-Contracting Module (ECM)

Contracting within the transport and logistics domain and complex international business relationships is currently a manual and time consuming process. The process begins with an order, i.e. the request of a customer to ship a good from place A to place B. This is followed by partner identification and qualification, partner bid development and bidding, bid evaluation and finally tentative partner selection. Once a tentative partner has been selected a contract is negotiated and agreed between various members of the contracting parties. The contract specifies all legal terms and conditions for the carriage of goods and SLA conditions such as: escalation processes for those occasions when problems arise, payment schedules and service level requirements. Unfortunately, all of this information is contained in a paper based document that is not generally available to the downstream individuals who are responsible for executing the contract and enforcing the agreements within.

The FInest E-Contracting Module (ECM) is being designed to address the highly manual nature of transport and logistics contracting and the problem of downstream transparency to contracted SLA conditions by exploiting solutions from e-contracting [7, 8]. It is important to remark that the legal terms and conditions of a contract are not in the focus of the ECM. The e-contracting module is envisioned as providing support for:

- Electronic model for representing the SLA attributes of T&L contracts (e.g., SLAs, pricing, escalation processes, etc.);
- On-line management and review of contracts with automatic notification of contract end dates and renegotiation time fences;
- Execution of semi-automated e-contracting selection (offering, bidding, choosing), establishment (negotiation and agreement), and management (reaction to deviations in the execution of established contracts).
- Integration of marketplaces to support (semi)-automated partner selection, bidding, and negotiation.

To facilitate this, the ECM is planned to encompass the following key architectural elements; their interaction as well as the primary interfaces of the ECM module is depicted in the high-level conceptual architecture shown in Figure 6:

- Contract Storage – Data repository for all established transport and logistics contracts, including a set of contract primitives, such as general attributes that characterize transport and logistics contracts.
- User Contract Demand Manager – Single interface where actors (human or electronic) interact with the ECM and inform it of the type of contract (e.g., blanket, spot, etc.) to be negotiated and established.
- Blanket, Capacity-based, and Tariff-Time-based Contract Managers – Elements responsible for assembling the electronic form of each respective type of contract (i.e., blanket, capacity-based, and tariff/time-based contracts), selection of partners (via an

auctioning or other process, according to the attributes of each type of contract) and electronic SLA part of the contract creation and storage.

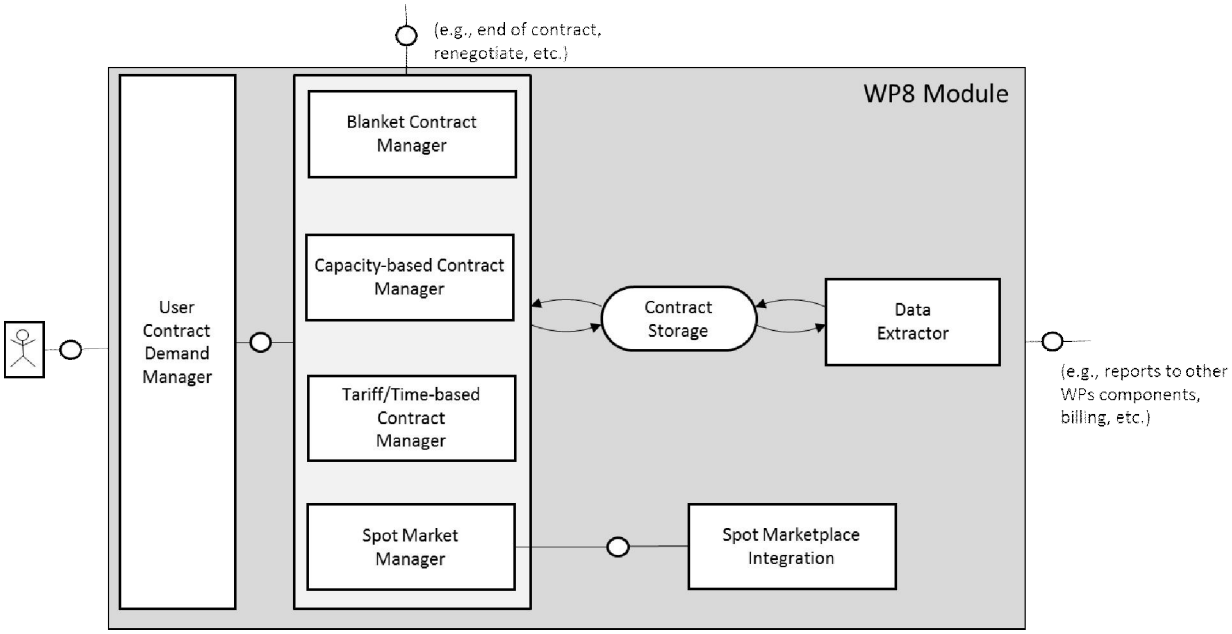

Figure 6. E-Contracting Module (ECM) conceptual architecture

- Spot Market Contract Manager – Element responsible for selecting qualified bidding partners for a spot contract, developing the spot contract and establishing the spot contract.
- Spot Marketplace Integration – A place for executing an auction process (offering, bidding, selecting, etc.). We do not need necessarily to define and create an entire marketplace support, but we need to provide the easy integration from the ECM and the already existent logistics marketplaces.
- Data Extractor – Element responsible for generating information about/from contracts for other external sources, which can be other core modules of the FInest platform (e.g. BCM or TPM) or other external systems wherein contract information are kept.

3.2.3 FInest Event Processing Module (EPM)

Event-driven architectures support applications that are reactive in nature, ones in which processing is triggered in response to events, contrary to traditional responsive applications, in which processing is triggered in response to an explicit request [5]. Furthermore, reactions are often based on situations identified by more than one event over periods of time and therefore processing events and inferring situations is a required capability in an event-driven architecture. In FInest, an event-driven architecture and an event processing module is employed for the purpose of end-to-end monitoring of logistics processes and to facilitate immediate and

proactive response to problems and potential deviations occurring during execution time. The functionality can be described at three levels.

- On the surface, event processing provides visibility into the current status of the logistics process: the location of a shipment, whether it is on a carrier or in a warehouse, whether or not it was customs-cleared, etc.
- Beyond the functionality of mere track-and-notify, event-processing employs rules that encapsulate specific logic applied to events. The basic functionality of rules exists in indicating whether or not the logistics process progresses as it should, or whether something has gone wrong.
- At a deeper level, events potentially provide insights regarding parts of the scenario that have not yet been reached; for example, stormy weather near a seaport may indicate that a ship carrying the managed containers will be delayed in entering the port. Security alerts at an airport may imply flight delays. Detecting those events relevant to the scenario at an early stage allows the system to respond to events *before* they occur, and thus to surface *proactive event-driven computing* functionality [6].

The Event Processing Module (EPM) can be characterized according to four elements, which are identified in the high-level conceptual architecture in shown in Figure 7 below:

- Event Sources – FInest differentiates between two types of event sources. The first type refers to various existing (but usually incompatible) systems. These include airport systems, sea freight systems employed by ports and vessels, scanning and tracking systems of packages, and others. The second type refers to sources that will be provided by the Future Internet infrastructure and will allow more accurate monitoring, as well as predictive capabilities; these include, for instance, RFID tags, smart cameras on roads and other smart items.
- Events – FInest also distinguishes between two types of events: the events emitted by existing sources (for example: delivery process completion, new order received) and events emitted from Internet of Things artefacts (for example: positions of a truck or reading an RFID tag of a container at a port upon arrival\discharge). While the latter must be defined and characterized in order to generate requirements from the Future Internet, existing events are described by domain sources such as *Cargo 2000*, which is an airline industry standard.
- Run-time Engine – The run-time engine exploits a set of rules to determine situations. Rules can either be permanent (for any scenario) or instantiated for a specific scenario, e.g., according to information about the execution of the transport plan (e.g., ETA for individual transport legs).

- Determined Situations – In FInest, the results of event processing are directed to a human interface, and to the BCM on the progress of running logistic processes and exception that need handling.

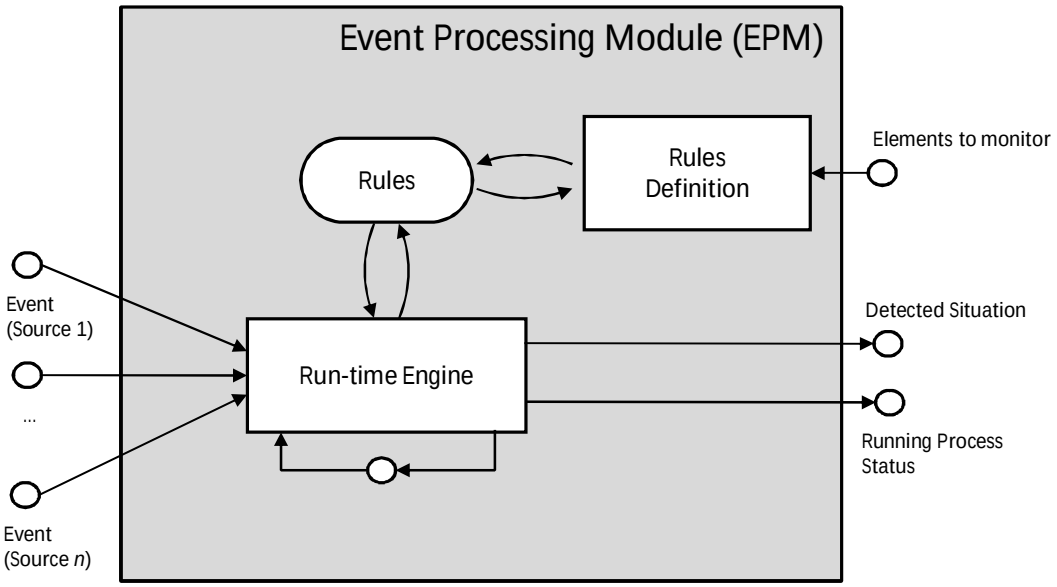

Figure 7. Event Processing Module (EPM) conceptual architecture

3.2.4 FInest Transport Planning Module (TPM)

Efficient and effective transport planning and re-planning is all about making sure that relevant information is available at the right time and place to support supply chain operations. Planning consists of resource and information requirements which are tightly linked to resource status, availability, and configurability. Resources can be found both inside a logistics service provider (LSP) and outside a LSP, e.g., in ports or customs agencies. The ability to incorporate or "wrap-in" this information into the planning process is essential. Today this is mainly done through manual inter-organizational collaboration processes [9, 10]. The aim of the FInest Transport Planning Module (TPM) is to overcome these business-critical shortcomings by making real-time information about resource status available across actors and organizational boundaries, which shall constitute a significant improvement in planning and optimization processes for international transport and logistics [11, 12].

Under consideration of the relationship with the other FInest Core Modules outlined above, the primary capability of TPM is to create an overall, operational transport plan for a multimodal transport chain handling goods by utilizing the relevant and most recent information that is available at the time of planning. The resulting transport plan encompasses the relevant information on the items to be transported as well as all details on the LSPs, transport legs, and required documents for executing the transport; this transport plan is then initiated within the BCM module (s.a.) for handling and controlling the execution. The second capability of the TPM is to support transport re-planning: an event related to an existing, ongoing transport

plan may lead to the need for re-planning (detected by the EPM module); after the re-planning is triggered (usually by human involvement due to business-relevance), the planning facilities of the TPM are used to revise the transport plan with respect to the current execution status of initial plan, which can be determined from the BCM module; the resulting new (or revised) plan is then (re-)instantiated within the BCM for continuing the execution.

In order to facilitate these capabilities, the TPM is planned to encompass the following key architectural elements; their interaction as well as the primary interfaces for the interaction and information exchange with the other FInest components is shown in Figure 8:

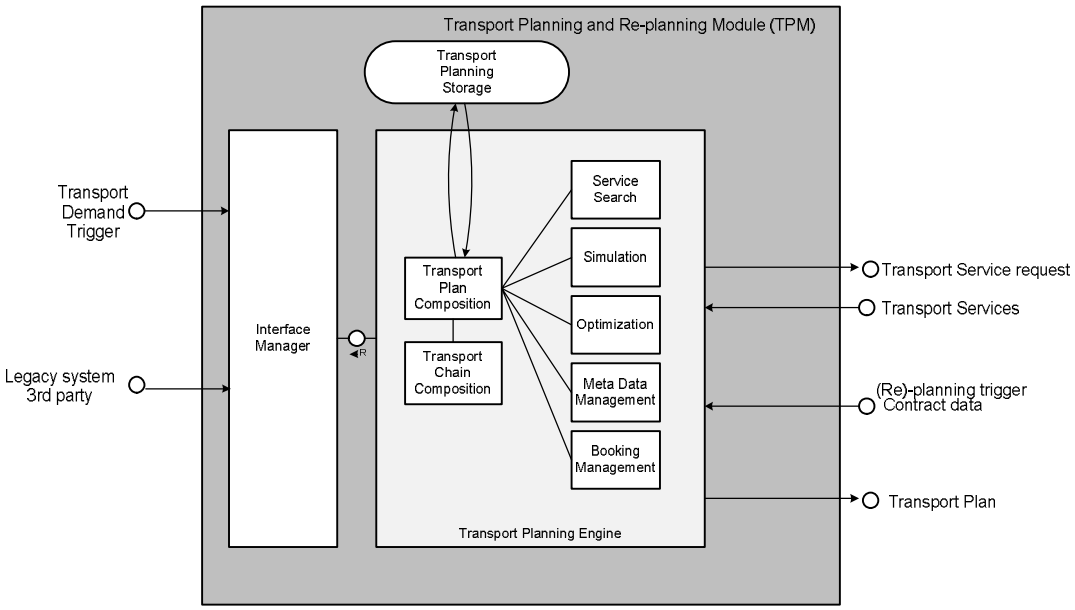

Figure 8: Conceptual Architecture of the Transport Planning Module (TPM)

Figure 8 shows an overview of the three components included in the TPM module (Interface Manager, Transport Planning Storage, Transport Planning Engine) and how they relate to triggers and input/output data which are described below:

- Interface Manager: Component to handle the interface between the planning components and the users.
- Transport Planning Storage: Object storage needed by TPM to store transport plans, meta data on transport plans etc.
- Transport Planning Engine: This component contains the following sub-components:
    - Transport Plan Composition
    - Transport Chain Composition
    - Service Search
    - Simulation

  - Optimization
  - Meta Data Management
  - Booking Management

- Other FInest modules with an interface to TPM: BCM (Business Collaboration Module) and ECM (E-Contracting Module). They are shown in the figure by having information exchange with TPM: Transport Plans are sent to BCM, and replanning events are received from BCM. Contract data is received from ECM, and transport service requests are sent to ECM.
- Transport Demand trigger: This includes both back end systems (third party transport provider systems) and end users accessing the planning module through a front-end. The planning process in TPM starts with an end user or third party system presenting their transport demand to TPM.
- Legacy system/Third party system: Some legacy systems, for instance for scheduling, transport service bookings, or optimization, may be useful for TPM, and thus be handled by the interface manager.

4. Technical Requirements Analysis

This section identifies initial technical requirements for realizing the FInest Platform as defined within the Initial Conceptual Architecture introduced above. We here focus on those elements that are (a) common to several core modules, such as service-based technologies or security & privacy management, and (b) needed for the technical realization of the FInest Platform as a whole, such as technologies for middleware management, user profile management, or graphical user interfaces technologies.

Considering the High Level Architecture introduced above in Section 3.1.2 and resulting from the iterative alignment of the technical work packages that design the FInest Core Modules (WP5 – WP8), the following technology areas have been determined to be relevant for realizing the FInest Platform as a whole and hence are inspected here:

(1) Middleware Management that is concerned with the techniques needed for enabling the extensibility of the FInest Platform as well as the deployment in cloud-environments and development of customized user applications on top
(2) Service-based System Integration as a promising technology for various parts of the FInest Platform, including the integration of external systems via the back-end, the interaction of the core modules, and the creation of user applications on top
(3) Security and Privacy Management that is concerned with ensuring secure, reliable, and trusted information exchange and storage, representing one of the essential requirements for the applicability of the FInest Platform in real-world business environments, and
(4) User Interface Technologies for facilitating the integrated user interfaces with support for various channels and devices as envisioned for the FInest Platform.

The following inspects each of these relevant technology areas, explicating its relevance for the FInest Platform, identifying the technical requirements for satisfying the desired functionalities that have been determined from interviews and iterative discussions from the domain experts, and outlining how these are planned to be addressed. Several of the identified requirements can most likely be satisfied by Generic Enablers that are planned to be provided by the FI-WARE project; this is discussed in more detail below in Section 5.

4.1. Middleware Management

One of the main business requirements expected from a collaboration and integration platform for the transport and logistics industry provided by WP1 is the need for rapid assembly of collaborative logistics networks that can efficiently and effectively execute international trading activities. Such collaboration includes a multitude of stakeholders as listed in Section 3.2.1 each with different roles, needs and purposes in the logistics process. Therefore the platform needs to sustain:

- Growing number of customers, shipments and participators in collaborations
- New and updated facilitations of parts of the logistics processes in a rapid manner (this may include new means of shipments by transport providers, new destinations, support of new types of shipments, offering new services such as customs support or insurance for special type of transport)
- Interoperability with stake holders' own systems such as e.g. communicating with the Port Community System (PCS) of a destination port notifying of a ship expected arrival or other necessary B2B interactions.

The central requirements for this are enlisted in the following table, organized as follows: *name* is the identifier of the requirement, the *functional requirements* describes the desirable features and relevance for transport and logistics that has been determined from the business requirements defined in WP1 and WP2 as well as iterative feedback from the domain experts, and the *technical requirements* identifies technologies, respectively technical building blocks that appear to be suitable to satisfy the desired functionality and have been identified by the ICT experts of the FInest consortium.

Table 1: Requirements on Middleware Management

	Name	Functional Requirements	Technical Requirements
1	**Scalability**	Need to support peaks in shipments and of their tracking, as well as growing number of different collaborations between stakeholders, of contractual agreements and of logistics processes to monitor and execute	a) IaaS Service Management – dynamic scalability
2	**Extendable**	All stakeholders need to be able to modify and extend their offerings, deploy new shipment capabilities (contract, shipment methods, destinations, prices etc.). Such capability is to be almost effort less – deployment and integration into a running Finest Platform at the level of click, drag, and point.	a) PaaS b) Application composition and mashup
3	**Core Modules deployable onto clouds**	Design and development of Finest core modules should be supported with methods and tools that will allow these modules to be hosted in the cloud	a) PaaS application description b) Tools that help develop applications and services deployable onto clouds
4	**Interoperability**	Enable interoperation between Finest (execution of logistics processes) with stakeholders' own systems at the required point in time	a) IaaS Service Management – federation and interoperability

Fortunately, the technical requirements to support such sustainability may be provided by state of the art middleware and cloud hosting technologies. Middleware Management covers the technologies required for developing and deploying the FInest core modules onto Future Internet / cloud-based platforms, for interoperation with stake holder's own system when required, as well as for supporting the process of deploying new offerings by stakeholders becoming part of the collaboration assembly and execution.

In particular, the following state-of-the-art, respectively emerging technologies appear to be potentially suitable candidates for satisfying the identified requirements:

- Platform as a Service (PaaS) - clients, typically application developers, follow a specific programming model and a standard set of technologies to develop applications and/or application components and then deploy them in a virtual application container or set of virtual application containers they rent. How these virtual application containers map into concrete runtime architecture is hidden to the application developer who does not need to have strong skills in system administration. However, this happens at the price of loosing part of the control on how the runtime architecture is designed. This model enables fast development and deployment of new applications and components. Usually, in addition to hosting, the PaaS providers also offer programming utilities and libraries that expedite development and encourage reuse.

- Infrastructure as a Service (IaaS) – offers virtualization capabilities that enable both high utilization and secure sharing of physical resources, and create a very flexible environment where logical computation processes are separated and independent from the physical infrastructure. IaaS enables secure sharing of physical resources through partitioning, support migration without limitations, and provide a holistic system-wide view and control of the infrastructure.

- Application Composition and Mashup - tools that empower users, from developers to domain experts to citizens without programming skills, to create and share in a crowdsourcing environment new added value applications and services adapted to their real needs, based on those offered from available business frameworks.

Such techniques are planned to be provided as Generic Enablers by the FI-WARE project, esp. in the context of the chapters on 'Cloud Hosting' and 'App / Service Ecosystem & Delivery (IoS)' as defined in [19]. Hence, for selecting the actual technologies to satisfy the identified technical requirements, we expose specific requests on Generic Enablers (cf. Section 5 for details on this), and will re-assess the suitability of the FI-WARE results in the next alignment iteration.

4.2. Service-based System Integration

Service-orientation allows building highly dynamic, distributed software systems, called service-oriented (or service-based) systems, and thus constitutes a key paradigm for flexible systems integration across organizational boundaries. A service-based system is realized by com-

posing individual software services. Software services contribute to a high degree of flexibility and interoperability by separating ownership, maintenance and operation from the use of the software. Service users do not need to acquire, deploy and run the individual piece of software, because they can access the functionality of that software from remote through the service's interface. The ownership, maintenance and operation of the software resides and is controlled by the service provider [8].

This shall allow future service-oriented systems to be composed from third party services that are accessible over the Internet and thus easily cross organizational boundaries, to be dynamically adapted to changed requirements and situations, and to be integrated with other Internet-connected capabilities and entities (such as connected devices) [17]. This allows for delivery models such as Software-as-a-Service [1, 2] or Cloud Applications that significantly progress from traditional application service provisioning (ASP) models.

The FInest platform consists of three main layers as introduced above in Section 3.1.2. The technical components and core modules that reside within these layers are envisioned to be flexibly interconnected by using service-oriented technology. Service-oriented technology facilitates interoperability, openness and extensibility through standard interfaces.

- The capabilities of the FInest Core Modules are envisioned to be offered as services (possibly delivered through cloud-based delivery models), which can be flexibly accessed and integrated through service interfaces depending on a user-specific instantiation of the FInest platform. This enables seamless communication and information exchange between those modules.
- The front-end layer of the FInest platform provides human-machine interfaces to the capabilities offered by FInest. Aiming to offer the core functionality in a service-based way (possibly flexibly rearranging and composing those domain-specific services), such a human-machine interface needs to be provisioned by dynamic and flexible technology, such as service mashup facilities.
- The back-end layer of the FInest platform, which provides access to and integration with legacy systems, third party services and Internet of Things (IoT) devices that may provide information during the transport lifecycle. Legacy system integration may be facilitated by exposing features of legacy systems as services (i.e., encapsulating legacy code with a service layer), or by offering access to legacy systems via the “Software as a Service” [1, 2] delivery model. In addition, the integration of external systems will allow automatically importing / exporting information to / from the FInest Platform (e.g., to access data stored in legacy data bases and systems).

Service-based system integration can be perceived as a certain kind of service composition, which considers both regular services (either within the organization or offered by third-parties) and legacy applications and things exposed as services (see above).

As discussed in [7], “a service composition is a combination of a set of services for achieving a certain purpose”. In this context, we can consider four types of service composition models:

- Service Orchestration: In service orchestration, a new service is created by combining several existing services in a process flow. The standard language for orchestrating Web services is WS-BPEL.
- Service Choreography: Service choreography defines the interaction protocol between services. Service choreographies are specified by means of languages such as WS-CDL or BPEL4Chor.
- Service Coordination: Service coordination models are needed when several services have to agree on the outcome of a distributed process by using a central coordinator and a coordination protocol. Service coordination models in the context of Web services can be defined using WS-Coordination.
- Service Assembly: Service assembly is performed during deployment of service based applications. A service assembly is a deployable artefact, which is deployed to an enterprise service bus. Service Component Architecture [(SCA), for instance,] implements the service assembly model. In addition, repositories and registries that provide a "yellow-page"-access to available services are employed to identify candidate services (possibly from various providers, both internal to the organization as well as external).

In addition, Semantic Web Service Compositions exist. This should not be considered following a separate type of composition model, but rather extending existing models and languages, in particular service orchestration, by use of semantic technologies in the context of Semantic Web services.

As can be seen from the above models, languages for describing services and their integration become key enablers in the setting of service-based system integration. Ideally, such a language should cover both technical and business requirements, and also come along with an integrated tool support for the provisioning, management, and consumption of services.

Table 2: Requirements on Service-based System Integration

	Name	Functional Requirements	Technical Requirements
1	**Service Orchestration**	Languages and tools for service orchestration	a) Service orchestration editor and execution engine
2	**Service Choreography**	Languages for modelling service choreographies and relating to service orchestrations (global vs. local perspective)	c) Choreography editor and analyser (matching orchestration to choreography)
3	**Service Coordination**	Languages and tools for service coordination (including monitoring and adaptation)	c) Service composition editor and engine for executing the service workflows d) SLA management and governance facilities

4	**Service Assembly**	Languages and tools for service description and deployment, as well as service search.	a) Service composition editor and engine for executing the service workflows b) Service and application mashup (targeted to front-end integration) c) Repository and/or registry facilities to store and retrieve service descriptions or parts of it (can be centralized or decentralized); in addition repository/registry needs to provide access to actual service endpoints and service provider information

The FI-WARE product vision as defined in [19], particularly the chapter on 'App / Service Ecosystem & Delivery (IoS)' plans to provide an integrated infrastructure and tool-support for service provisioning and management. This, among others, plans to provide Generic Enablers that appear to be a promising basis for satisfying the identified technical requirements. Hence, we expose specific requests on Generic Enablers for this technology area (see Section 5 below), and will assess the suitability of the FI-WARE results in the next alignment iteration.

4.3. Security and Privacy Management

Sophisticated security and privacy management techniques are a pre-requisite for the industrial applicability of the FInest Platform in real-world environments: in general, enterprises will not employ a technology in productive environments with business-critical information if the information confidentially as well as secure and reliable data storage and transfer cannot be guaranteed; this becomes particularly relevant for distributed and cloud-based environments where information shall be shared across business partners.

Access control, secure communications, monitoring and event management are the key concerns for the security: security solutions like firewall, IPS, anomaly detections systems and malware detection systems are important for defence for attacks; VPN technologies will supply the secure connections to the systems; 2-factor authentication will be more secure authentication mechanism comparing the others. Security monitoring is the first step towards understanding the real security state of a future internet environment and, hence, towards realising the execution of services with desired security behaviour and detection of potential attacks or non-authorized usage. For Perimeter Security, next generation firewalls can be used; IPS and Network Anomaly Detection systems will be useful for detecting known and unknown attacks; Database and Web security are also considered for specific attacks.

Table 3: Requirements for Security and Privacy Management

	Name	Functional Requirements	Technical Requirements
1	**VPN - Virtual Private Network for Communications**	Secure Communication will be needed for web connections or IPSec tunnel for layer 3 communications	a) SSL VPN technologies will be suitable for secure connections. It should be platform independent like mobile, desktop, etc b) It should be integrated with AAA systems like radius or single sign on solutions) c) User role mapping and policies can be applied through sslvpn for better control and security
2	**Identity Management**	All the users must use their own accounts and when the user's business role changed his/her authorization must be changed on systems automatically. Creating new accounts or disabling account could be done from centralized system.	a) The target resources will be connected with web based Identity Management tool b) IdM must be web-based application c) End user can login and change password or non-critical attributes individually d) There will be a trusted resource for employees and IdM will get information from these systems and sets the values to target system individually. e) Key users or local administrator could give users some authorization at their own resources and/or organizations
3	**Access Control**	Access Control is very important to reach data. Key points as below -Two factor Authentication methods will supply better security control - Single Sign on solutions will be better for managing and accounting purposes - PKI infrastructure can be applied according to the needs.	a) Soft Token solutions can be implemented for mobile or desktop users. Hence they will provide OTP (one time passwords) and their pin code with user names. b) Single Sign on solutions will be more comfortable for users. After authentication, users will not need again authentication for other systems c) Support for extension of the UI for the features & functionalities provided by additional

			technical modules or components d) For PKI infrastructure a trust center will be needed for users. Trust Center will be responsible for key exchange mechanisms and keeping them in a secure way. Digital Signature can be used for trust relationship.
4	**Security Monitoring & Event Management**	Analyzing the events is important for the cyber attacks. Security monitoring is focused essentially on monitoring alarms from network equipment, systems and security sensors	a) Events should be stored and generate alarms for critical issues b) Graphical dashboard show the analysis and reporting of the modules c) the availability of digital forensic evidence models and tools will provide a forensic solution to analyse abnormal behaviour,
5	**Security Enablers**	Next Generation Firewall IPS Antivirus – Malware detection Network Anomaly Detections Database and Web Application Firewall	a) Next Generation Firewall will have ability of the application and identity awareness b) IPS will defend known attacks by using signatures c) Antivirus , Malware detection is important d) Network Anomaly Detections can reduce zero day attacks and new malware e) Database and Web Application firewall solutions

The following table outlines how the single technical requirements for security and privacy management are planned to be addressed. In addition to this, we expose specific requests on Generic Enablers on the chapter 'Security' as described in the FI-WARE product vision [19], which plans to develop an integrated framework for security, privacy, and trust management on the Future Internet and hence will expectably provide at least some technical solutions for satisfying the technical requirements for FInest as identified above.

Table 4: Plan for Addressing Requirements on Security and Privacy Management

	Name	Addressed by
1	VPN - Virtual Private Network for Communications	SSL VPN can be implemented like a gateway solution to reach DMZ servers or internal servers. On some mobile solutions (android, iPhone, windows mobile) IPSec-tunnelling is possible.
2	Identity Management	Web based centralized identity management tool can be used.
3	Access Control	2-Factor authentication can be supplied in various ways, e.g. by hardware tokens or by soft tokens for mobile devices. PKI infrastructure is also an option for better security.
4	Security Monitoring & Event Management	Event Management systems can collect the logs from firewall, IPS, SSL VPN, identity managements and etc. It can correlate the data but for forensic investigations, it needs raw data. Also automatic actions are not possible for most of the vendors. FI-WARE will supply these monitoring and event management systems.
5	Security Enablers	On the FI-WARE platform, integrated solutions like firewall, IPS, malware and antivirus detection systems will be needed. On cloud environments, third party solutions can be also used.

4.4. User Interface Technologies

The fourth relevant technology area is concerned with technologies and frameworks for user interfaces. This relates to the Front-End of the FInest Platform, where the aim is to provide an integrated user interface for all core modules that offers a 'single point of access look & feel' with customizable views for each individual user. In addition, accessibility via different devices (desktop, web, mobile) and different channels (that is technical access points) shall be supported in order to allow the various roles to access the FInest Platform and utilize its functionalities from different environments (locality, technical infrastructure, etc.). For this, the following table determines central functional and technical requirements on the integrated user interface for the FInest platform.

Table 5: Technical Requirements for User Interfaces

	Capability	Functional Requirements	Technical Requirements
1	**Access via Multiple Channels and Devices**	The individual users of the FInest Platform are located in different environments, and hence use different channels and devices for accessing and working with the FInest Platform and technologies. Examples are e.g. (1) a transport organizer located in a	a) User Interface Technologies to support access to the FInest Platform and usage of its features by different devices (desktop, web, mobile) b) Access to the FInest Platform via different channels (techni-

		operation center of a LSP, using a desktop / laptop with broadband Internet connection (2) the loading manager at a pick-up point or port using mobile devices, or (3) carrier personnel such as truck drivers or a ship crew accessing the FInest Platform via different devices in remote areas	cal interfaces and networks, e.g. broadband Internet access, mobile, etc.)
2	**'Single Point of Access'**	The user interface shall provide a 'single point of access' for all features, functionalities, and services of the FInest Platform (or of an customized application built on top of it), independent of which technical modules or components are actual used to solve a specific user action (e.g. placement of a new order, perform transport planning, invite business partner, request for new contracts, etc.)	a) Integrated User Interface = central access point to all features / functionalities / capabilities provided by the FInest Platform b) De-coupled 'business logic' for invoking the specific technical components to solve a particular user action c) Support for extension of the UI for the features & functionalities provided by additional technical modules or components
3	**Uniform Look & Feel**	For better user experience, the integrated FInest User Interface shall provide a uniform look & feel	The graphical user interfaces for each functionality / component / module shall a) Be developed using the same user interface technologies b) Follow common UI design patterns for similar user experience (corporate design)
4	**Customized Views for Individual Users**	Each individual user shall have an personalized view on the FInest Platform that is adapted to the personal needs and the given access rights to data, information, and processes	a) Integrated Access Control and management for users and roles b) Support for individual personalization, customization, and extension of the FInest User Interface

In contrast to the previously discussed technology areas, the FI-WARE product vision as presented in [19] does not plan to provide a comprehensive set of Generic Enablers for user interface of Future Internet service and applications. Hence, the plan for satisfying the identified technical requirements consists of the following parts: identification of potential candidate technologies (provided in the Appendix, see Section8.1), definition of governance models for ensuring the uniform look and feel, and the integration of other technical building blocks. The following table explains this in detail.

Table 6: Plan for Addressing Requirements on User Interface Technologies

	Name	Addressed by
1	Access via Multiple Channels and Devices	a) Choosing a GUI technology for the presentation layer which is available at the most platforms and devices; for this, a initial examination of potential candidate technologies in provided in the Appendix (see Section 8.1) b) Using the Generic Enabler "Interface to Network & Devices (I2ND)" by FI-WARE which enables communication via multiple channels and via the cloud
2	'Single Point of Access'	All sub-requirements are mainly 'design tasks' and cannot be supported by a technology at all. Only 2b is supported by some GUI technologies but can also be achieved by a proper platform architecture and design.
3	Uniform Look & Feel	a) Agreeing on one GUI technology used for the entire presentation layer of the FInest platform will support this but will not guarantee an uniform look and feel b) Additionally this is subject to 'procedures & guidelines' for the technical WPs and cannot be supported by a GUI technology
4	Customized Views for Individual Users	a) Will be facilitated by integrating Security & Privacy Mgt techniques investigated in above in Section 4.3 above into the FInest Integrated UI b) Like 2c) this is a 'design tasks' and cannot be supported by a technology at all, but by a well designed UI

5. Initial Generic Enablers Requests

After having presented the initial conceptual architecture and the identification of technical requirements for the overall FInest Platform, this section provides the initial version of requests on Generic Enablers which the FInest project, as a use case project, demands from the FI-WARE project that develops the Future Internet Core Platform.

As already stated in the preceding section, the FInest platform shall be designed as a domain-specific extension of the FI-WARE platform, therewith realizing the overall aim of the FI PPP program. This means that the FInest platform will be a 'collection of enablers' – encompassing both Generic Enablers from FI-WARE as well as the domain-specific ones developed in FInest – which are deployed (or at least are deployable) in a cloud environment. In addition, all these 'enablers' are expected to use the same technical environment and comply with the same rules for interoperability (which are also expected to be defined by FI-WARE). In order to identify and agree on the Generic Enablers, the FI PPP has set up the Architecture Board as a part of the program-wide coordination and procedures (see Deliverable D9.1 [20] for further details), which is responsible for defining the procedures and tools for the interaction between FI-WARE and the use case projects. As the basis for defining the initial requests on Generic Enablers by the FInest project, we shall recapture the current status on the procedures, tools, and templates defined by the FI PPP Architecture Board below in Section 5.1.

After that, Section 5.2 presents the complete list of initial requests on Generic Enablers by the FInest project. This covers inputs from all technical work packages, i.e. from WP3 on the technology areas that are relevant for the FInest Platform as a whole, and from WP5 – WP8 with respect to the specific technical requirements identified for each of the core modules.

5.1. Procedures and Tools for Generic Enabler Requests

The overall interaction between the FInest project and the FI-WARE project follows the agile process elements, which have been defined by the FI PPP Architecture Board.[5] The following explains the aspects relevant for the preparation of initial request on Generic Enablers.

In general, there are three different levels of granularity for expressing Generic Enabler Requests (i.e., entries for the FI-WARE backlog):

- "Agile Themes: A Theme is a top-level objective that may span projects and products. Themes may be broken down into sub-themes, which are more likely to be product-specific. At its most granular form, a Theme may be an Epic. [...]
- Agile Epics: An Agile Epic is a group of related User Stories. [...].

[5] The agile methodology appliyed in FI-WARE (and hence in the FI PPP Architecture Board) is presented here: https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Agile_Development_Methodology; still under construction at the time of writing.

- Agile User Stories: A User story is an Independent, Negotiable, Valuable, Estimatable, Small, Testable requirement ("INVEST Acronym"). Despite being Independent i.e. they have no direct dependencies with [other] requirements, User stories may be clustered into Epics when represented on a Product Roadmap. User Stories are great for Development Teams and Product Managers as they are easy to understand, discuss and prioritise – they are more commonly used at Sprint-level. User Stories will often be broken down into Tasks during the Sprint Planning Process – that is unless the stories are small enough to consume on their own." [15]

In order to gather and properly process the requests on Generic Enablers that are expected from the FI PPP Use Case projects, the Architecture Board has defined a detailed template along with illustrative examples for the FI-WARE backlog entries; the template for the Generic Enabler Requests is provided in Appendix (see Section 8.2).

In the context of the FI-PPP, FI-WARE has provided the following clarification of these levels of granularity:"[…] there is no black / white frontier between things nor a mathematical formula one can apply and then determine when something should be considered an Epic rather than a User Story, for instance. […] User Stories comply with "INVEST" properties […]. […] in general:

- User Stories have to be something "Small", as to be affordable in one Sprint.
- Sprints should be of a *maximum* of two months in FI-WARE, unitary tests included.
- In general, anything beyond a single sprint should be considered as Epic.
- A User Story should be detailed enough - "I understand what I want precise enough to define a test for it" ("Testable" property)
- User stories should be "Estimatable", that is containing enough information enabling a developer to make a resource estimation for design, develop, and test (in 1 sprint).
- There is neither the need to cover all details nor to have everything finalized; there should be details that may be worked out while developing. [16]

At the time of writing, it has been understood by the FI PPP Architecture Board meetings that UC projects (and thus FInest) are not expected to provide GE requests on the granularity level of User Stories, but that it would be sufficient to provide them on the level of Epics.

5.2. Initial Requests on Generic Enablers by FInest

The following presents the complete list of initial requests on Generic Enablers by the FInest project. This represents the initial results of the tasks for identification of Generic Enablers from several work packages (T3.3, T5.3, T6.3, T6.3, and T8.3); it was decided to provide this here in an integrated manner (cf. task T9.1) in order to provide a single point of reference and also to avoid duplication. For collecting and aligning the initial list of Generic Enablers for the FInest project, the following process has been conducted:

- Each technical work package (i.e., WP5 - WP8 and WP3) identified the specific technical requirements, and, with respect to this, prepared a list of potentially relevant Generic Enablers by examining the FI-WARE product vision presented in [19]
- In a face-2-face meeting, all WP leaders and WP members have come together to jointly identify potentially relevant Generic Enablers, with respect to the identified technical requirements. During this process it was justified whether a technical requirement (or parts thereof) can be realized by means of some more generic technology, or whether it requires clearly domain-specific solutions. In addition to the list of technical requirements, this process was guided by the FI-WARE product vision [19]
- Based on the agreed list of initial generic enablers, each of those was refined and specified according to the procedures and format defined by the FI PPP Architecture Board.

Table 7 below provides an overview on the initial Generic Enabler Requests defined by the FInest project at M6 of the FI PPP program (= M6 of the FInest project). We here content ourselves with an executive summary that indicates the name, the related chapter and Generic Enablers as defined the FI-WARE product vision [19], and the allocation within the FInest Platform and the associated work package. The detailed definitions are provided on the public website of the FInest project: http://www.finest-ppp.eu/index.php/project-results/generic-enablers; that online list is planned to become a 'living document' where the requests are refined during the duration of the project; it might be joined or replaced by the FI-WARE backlog system that is set-up at the time of writing.

Analyzing the list of initial request on Generic Enablers, we can observe the following central aspects on the demands of the FInest project on the Future Internet Core Platform

- The FInest project demands Generic Enablers from all chapters that are defined within the FI-WARE product vision presented in [19]
- The FI-WARE chapters that appear to be most important for FInest are:
  a) App / Service Ecosystem & Delivery (IoS)
  b) Security
  c) Cloud Hosting
  d) Data / Context Management (selected GEs)
- The Generic Enabler Requests enlisted below can be grouped as follows:
  a) The vast majority of the initial requests on Generic Enablers represents requests for refinement or extension of Generic Enablers that have already been described in the FI-WARE product vision [19]

b) The second, significantly smaller group covers requests for new Generic Enablers, i.e. features or technologies that are needed for realizing the FInest Platform and may also be demanded in other application domains

c) The smallest group present 'declarations of interest', meaning that a Generic Enabler as already defined in the FI-WARE product vision [19] appears to be useful for FInest without demands of refinement or extension (at this point in time).

Table 7: Overview FInest M6 Requests for Generic Enablers

No.	Name	FI-WARE Generic Enabler	FInest Component	FInest WP
FI-WARE CHAPTER: App / Service Ecosystem & Delivery (IoS)				
1	Workflow Execution Engine	Service Orchestration Engine	BCM	WP5
2	Widget Platform & Infrastructure	NEW	Integrated User Interfaces	WP3
3	Application composition and connectivity	Composition & Mashup	Middleware Management	WP3
4	Graphical Presentation of Workflow Execution	Composition & Mashup	TPM	WP7
5	Languages and tools for service orchestration	Composition editor ; Application mashup editor; Service composition editor	Overall	WP3
6	Languages for modelling service choreographies and relating to ser-vice orchestrations (global vs. local perspective)	Service orchestration engine	Overall	WP3
7	Languages and tools for service coordination (including monitoring and adaptation)	Execution engine; Mashup execution engine; SLA Management	Overall	WP3
8	Languages and tools for service description and deployment, as well as service search.	Aggregator repository; USDL Repository; USDL Registry	Overall	WP3
9	Financial clearing services	Financial services	Overall	WP3, 5, 6, 7, 8
FI-WARE CHAPTER: Internet of Things (IoT) Services Enablement				
1	IoT event subscription	Data Handling	Backend	WP3
FI-WARE CHAPTER: Data / Context Management				
1	Big Data Analysis at Runtime	Big Data Analysis	BCM	WP5

2	Query Broker	Query Broker	BCM	WP5
3	Event processing and prediction	CEP	EPM	WP6
4	Capture and collect events	Massive Data Gathering	EPM	WP6
5	Disseminate derived events	Pub\Sub	EPM	WP6
6	Optimization Models and Execution	Big Data Analysis	TPM	WP7
7	Non-repudiation services	?	Architecture	WP3
FI-WARE CHAPTER: Cloud Hosting				
1	Trusted Object Storage	Object Storage	BCM	WP5
2	Dynamic Scale	IaaS Service Management	Middleware Management	WP3
3	Systems Interoperability	IaaS Service Management	Middleware Management	WP3
4	Application deployment & management	PaaS Management	Middleware Management	WP4
5	SLA & Contracts storage and retrieval	Object Storage	ECM	WP8
FI-WARE CHAPTER: Interface to Network & Devices (I2ND)				
1	Interface to Networks and Devices (I2ND) for Future Internet Applica-tions	CDI ,Cloud Edge , NetIC, S3C	Integrated User Interfaces	WP3
FI-WARE CHAPTER: Security				
1	Virtual Private Network for Communications	Security Enablers	Overall (secure information transfer)	WP3
2	Identity Management	Identity Management	User Management & Access Control	WP3
3	Access Control	Identity Management; Identify Mixer	User Management & Access Control	WP3
4	Security Monitoring & Event Management	Security Monitoring	Overall	WP3
5	Basic Security Enablers	Security Enablers	Overall	WP3
Additional GE Requests (not associated to FI-WARE Chapter yet)				
1	Configurable non-repudiation mechanisms	--	Overall	WP3
2	Financial clearing services	--	Overall	WP3

6. Conclusions and Outlook

As the first deliverable of work package WP3 "Solution Design and Technical Architecture", this report has presented the initial conceptual architecture of the envisioned FInest Platform that shall facilitate the optimization of collaboration and integration within transport and logistics business networks along with the identification of functional and technical requirements for the central components and building blocks. In addition, the procedures and tools for aligning and coordinating the R&D work among the work packages concerned with the design and technical specification of the FInest Platform have been defined.

The overall aim of the FInest project is to develop a Future Internet enabled ICT platform that shall facilitate optimizing the collaboration and integration within international transport and logistics business networks. As the initial conceptual architecture for this, we have defined a high-level architecture that identifies three layers (the 'Front-End' encompassing integrated user interfaces and security-based access control to the FInest Platform; the 'Core Modules' that provide domain-specific capabilities for improving the collaboration and integration throughout transport and logistics business networks, and the 'Back-End' that facilities the integration of existing and legacy business systems for automated data import and export).

While the platform shall be extensible by adding additional functional modules, the FInest project will design the following 4 Core Modules in correspondence to the technical work packages: the 'Business Collaboration Module (BCM)' that captures all information that are relevant for efficiently executing logistics processes in an collaborative manner (WP5); the 'Event Processing Module (EPM)' that monitors events relevant for planning and executing logistics processes and triggers respective activities (WP6); the 'Transport Planning Module' that supports semi-automated planning and re-planning of logistics processes with the relevant information at real time (WP7), and the 'E-Contracting Module' that provides support for establishment and management of transport contracts and allows programmatic access to transport and logistics contracts (WP8). We here have defined the capabilities as well as the interfaces for information exchange of the modules with respect to the business processes that shall be supported, therewith ensuing the overall interoperability; for each module, the initial design along with the detailed analysis of technical requirements and a state-of-the-art analysis is provided in the first deliverables of the respective work packages (cf. D5.1 – D8.1).

We further have determined the following technology areas to be relevant for the FInest Platform as a whole: (1) *Middleware Management* for enabling the extensibility of the FInest Platform as well as the deployment in cloud-environments and development of customized user applications on top, (2) *Service-based System Integration* as a promising technology for various parts of the FInest Platform, (3) *Security and Privacy Management* for ensuring secure, reliable, and trusted information exchange and storage, representing one of the essential requirements for the applicability of the FInest Platform in real-world business environments, and, (4) *User Interface Technologies* for facilitating the integrated user interfaces with support for various channels and devices as envisioned for the FInest Platform. For each of these, we

have identified the technical requirements for satisfying the functionalities that have been determined by the domain experts, and outlined how these are planned to be addressed. For most of these technologies, we expect the basic facilities to be provided in form of Generic Enablers by the FI-WARE project. For this, we have elaborated a collection of requests on Generic Enablers that will be provided as input to the FI-WARE backlog via the tools, procedures, and templates defined by the Architecture Board. The list of more than 30 requests on Generic Enablers presents the aggregated requests gathered from all technical work packages (i.e. WP3 and WP5 – WP8); we provide this in an integrated manner in order to provide a single point of reference and also to avoid duplication.

In addition to work as defined in the DoW, we also have refined the relationship and interactions of the work packages in the FInest project and defined the procedures and tools for aligning and coordinating the R&D efforts in the technical work packages: the business requirements from WP1 as well as the concrete use case scenarios from WP2 will directly be delivered to the technical work packages: each of WP5 – WP8 is responsible for a thorough state-of-the-art analysis, while the role of WP3 is to coordinate the process and to consolidate the outcomes; this presents an additional key activity of WP3 with respect to the FInest DoW. As one of the first decisions for the coordination, we have introduced Technical Architecture Modeling (TAM) as the technique to be used throughout the project for the design and detailed specification of the FInest Platform and its core modules.

With this, the present deliverable has presented the initial version of the FInest Platform, which – in accordance to the iterative methodology applied within the FInest project – shall be refined and further developed throughout the following milestones. The primary objectives for the next milestone (M12) of WP3 are

- Continue the iterative refinement of the FInest Platform with its components and core modules with respect to the business requirements identified in WP1 and the concrete use cases elaborated in WP2
- Work towards an initial Technical Design of the FInest Platform
- Continue selection of the technology baseline and examination of potential candidates for technical realization with special attention to the Generic Enablers developed within the FI-WARE project.

7. References

1. Turner, M., Budgen, D., Brereton, P. Turning Software into a Service, Computer, 36(10), 2003.
2. Mietzner, R., Metzger, A., Leymann, F., Pohl, K. Variability modeling to support customization and deployment of multi-tenant-aware Software as a Service applications, In: in Proceedings of the ICSE 2009 Workshop on Principles of Engineering Service Oriented Systems (PESOS), 2009.
3. Meijler T.D. et al. Coordinating Variable Collaboration Processes in Logistics, In: 13th International Conference on Modern Information Technology in the Innovation Processes of Industrial Enterprises, Trondheim, Norway, 2011.
4. Hull, R. et al. Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles, In: 7th International Workshop WS-FM 2010, Hoboken, USA, 2010.
5. Etzion O., Niblett, P. Event Processing in Action, Manning, 2010.
6. Engel, Y., Etzion, O. Towards Proactive Event-Driven Computing, In: 5th ACM International Conference on Distributed Event-Based Systems (DEBS), 2011
7. Papazoglou, M., Pohl, K., Parkin, M., Metzger, A. (Eds.) Service Research Challenges and Solutions for the Future Internet: S-Cube – Towards Mechanisms and Methods for Engineering, Managing, and Adapting Service-Based Systems. Springer, 2010.
8. Nitto, E. D., Ghezzi, C., Metzger, A., Papazoglou, P., Pohl, K. A journey to highly dynamic, self-adaptive service-based applications, Autom. Softw. Eng., 15(3-4), 2008.
9. Stevens, G.C. Integrating the Supply Chain, International Journal of Physical Distribution & Logistics Management, 19(8), 1993.
10. Al-Mashari, M., Al-Mudimigh, A., Zairi, M. Enterprise resource planning: A taxonomy of critical factors, European Journal of Operational Research, 146(2), 2003.
11. Fjørtoft, K. et al. MIS - Identification and organization of MIS users and processes, Delivery A, MARINTEK, 2010.
12. Sleire, H., Wahl, A.M. Integrated Planning - One road to reach Integrated Operations. SPE Bergen Conference, 2008.
13. The FREIGHTWISE project; http://www.freightwise.info/cms/
14. The TCMS project; http://www.sintef.no/Home/MARINTEK/Software-developed-at-MARINTEK/TCMS/
15. Agile 101: The Difference Between Agile Themes, Epics and User Stories; http://agile101.net/2009/08/10/the-difference-between-agile-themes-epics-and-user-stories/
16. FI-WARE Wiki. http://wiki.fi-ware.eu/index.php/FI-WARE_Agile_Development_Methodology
17. Tselentis, G.; Domingue, J.;Galis, A.; Gavras, A.; Hausheer, D.:"Towards the Future Internet: A European Research Perspective". Amsterdam, The Netherlands: IOS Press, 2009.
18. Franklin, R.; Metzger, A.; Stollberg, M.; Engel, Y.; Fjørtoft, K.; Fleischhauer, R.; Marquezan, C.; Ramstad, L. S.: "Future Internet technology for the future of transport and logistics," in ServiceWave 2011, Future Internet PPP Track, ser. LNCS, A. Zisman, I. Llorente, M. Surridge, W. Abramowicz, and J. Vayssiére, Eds. Springer, 2011.
19. Hierro, J.J. (ed.): "FI-WARE High-Level Description (Product Vision)". FI-WARE Report, 15. August 2011.
20. Stollberg, M. (ed.): "FI PPP Alignment Plan". FInest Deliverable D9.1, 30. June 2011.

8. Appendices

8.1. Initial Examination on Graphical User Interface Technologies

(A) Desktop Technologies

(A.1) Windows Presentation Foundation (WPF)

WPF is a graphics framework and part of Microsoft's .NET-Framework and can be considered to replace the predecessor Windows Forms. It uses the Extensible Application Markup Language (XAML) for declarative user interface descriptions. WPF Applications can be built using different programming languages like C# and Visual Basic, for example [1]. With Silverlight, a web version of WPF is also available which runs in browsers.

Identification of Capabilities

The benefits are mainly the concept of decoupling the user interface and business logic in all applications by default, so both can be maintained without affecting each other. In addition, WPF supports graphics acceleration which means that the GPU takes over responsibility for calculating visual objects to disburden the CPU. WPF is well developed and established on Windows operating systems [1] but not supported on Unix-based systems. Even Windows Phone does support Silverlight instead of WPF [3]. This is a major downside.

Available on platforms:

Operating System	**Available**
Windows	Y
Linux	N
OSX	N
Android	N
Symbian	N
iOS	N
BlackBerry OS	N
Windows Phone	N

Specific References

[1] 2011. Microsoft Corporation. *Introduction to WPF*. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa970268.aspx

[2] 2011. Microsoft Corporation. *Silverlight Overview*. [Online]. Available:http://msdn.microsoft.com/en-us/library/bb404700%28v=VS.95%29.aspx

[3] 2011. Microsoft Corporation. *How to: Create Your First Silverlight Application for Windows Phone*. [Online]. Available:http://msdn.microsoft.com/en-us/library/ff402526%28v=vs.92%29.aspx

(A.2) Java Desktop Applications

The main GUI widget toolkits for the programming language Java, which contain all basic UI elements needed for implementing form-based GUIs, are Swing and SWT. The existence of two different toolkits is determined by historical performance reasons. Swing provides platform-independent UI-components while SWT provides native implemented components for each platform. The latter has been invented because the former had been considered to be slow. Because of today's high hardware capabilities the performance differences are not that important anymore. While Swing and SWT are for development of form-based GUIs, JavaFX has been developed for multimedia-based GUIs (e.g. for painting programs or charts) [1].

Identification of Capabilities

A special capability of Java programs is that they can run ether as desktop applications or as applets (in-browser applications). Normally only minor efforts are necessary to transform the former into the latter. All Java programs (including applets) require at least the Java Runtime Environment for execution.

However, as Java is a platform-independent programming language, desktop applications written in Java run on most operating systems except of some mobile operation systems. This is a main benefit of Java applications.

In contrast, Java applications may have an insignificantly slower performance then applications written in native programming languages. One of the main downsides of Java in terms of serving as frontend-technology is the insufficient tool support for the creation of user interfaces. GUI designers are often only support the three UI frameworks Swing, SWT and the predecessor AWT or even just one of them. In addition the majority of the designers are commercial programs [3], [4].

Available on platforms (Desktop Applications, not Applets):

Operating System	**Available**
Windows	Y
Linux	Y
OSX	Y
Android	Y
Symbian	Y
iOS	**N**
BlackBerry OS	Y
Windows Phone	**N**

Specific References

[1] 2006. Feigenbaum, Barry. *SWT, Swing or AWT: Which is right for you?*.[Online]. Available: http://www.ibm.com/developerworks/java/library/os-swingswt/index.html

[2] 2011. Castillo, Cindy. *What is JavaFX?*.[Online]. Available: http://download.oracle.com/javafx/2.0/overview/jfxpub-overview.htm

[3] 2007. Steppan, Bernhard.*GUI-Builder für Eclipse*. [Online]. Available: http://wiki.computerwoche.de/doku.php/programmierung/gui-builder_fuer_eclipse#vergleich_der_features

[4] 2011. Oracle Corporation. *Swing GUI Builder*. [Online]. Available: http://netbeans.org/features/java/swing.html

(B) Web Application Technologies

(B.1) Silverlight

Silverlight (previously known as WPF/E) is a subset of WPF for building web applications, his means a Silverlight application is located on a server and will be downloaded by a browser, which also runs it.

Identification of Capabilities

Silverlight has similar problems like Flash and Java Applets. The applications need to be downloaded first which is much slower than the interpretation of a plain HTML website, for example. Also Silverlight requires – like Flash – a browser plug-in, which is not available on all platforms and devices.

In contrast to WPF, Silverlight does also run on OSX [1]; support for iOS has been announced for the next Silverlight version [2]. Like WPF Silverlight does support graphics acceleration. Thus, compared to Java applets Silverlight performs much faster. With Moonlight also an open source implementation of Silverlight is available which also runs on Unix-based systems [3], but this not future-proof and therefore will not be addressed here.

Available on platforms:

Operating System	**Available**
Windows	Y
Linux	N
OSX	Y
Android	N
Symbian	N
iOS	N
BlackBerry OS	N
Windows Phone	Y

Specific References

[1] 2011. Microsoft Corporation. Get Microsoft Silverlight. [Online]. Available: http://www.microsoft.com/getsilverlight/Get-Started/Install/Default.aspx

[2] 2010. Viticci, Federico. *Microsoft's Silverlight To Gain iOS Support in 2011*. [Online]. Available: http://www.macstories.net/news/microsofts-silverlight-to-gain-ios-support-in-2011/

[3] 2009. Novell Inc.*Moonlight*.[Online]. Available: http://www.mono-project.com/Moonlight

(B.2) Java Applets

In contrast to Java desktop applications applets are Java programs which can run in browsers. This makes Java applications available on the fly at client side, where they will be executed [1]. Like in Java desktop applications SWT, Swing and JavaFX can be used to build GUIs.

Identification of Capabilities

On one hand, a benefit of Java applets is that almost the whole functionalities of Java can be used which makes Java applets almost as powerful as desktop applications. On the other hand, applets need to be downloaded before they can be executed and they require the Java Runtime Environment (JRE) installed on the device and a browser plug-in. The first of those facts is critical because users are used to work with web content after a load time of maximum 2 seconds per website and applets can load up to several minutes depending on the applet size. This might be the reason why today only few java applets are available on the World Wide Web. Additionally, the need of a browser plug-in and the JRE disallows running applets on all kinds of mobile devices. [2]

Available on platforms:

Operating System	**Available**
Windows	Y
Linux	Y
OSX	Y
Android	Y
Symbian	Y
iOS	**N**
BlackBerry OS	Y
Windows Phone	**N**

Specific References

[1] 2010. Oracle Corporation. *Applets*.[Online]. Available:http://java.sun.com/applets/

[2] 2011. Carranza, Joel. *Java Applets loading at snail's pace – Answer 1*. [Online]. Available:http://stackoverflow.com/questions/457515/java-applets-loading-at-snails-pace

(B.3) HTML5

The new version of HTML is called HTML 5.0 and includes by definition CSS 2.0 and JavaScript. It supports a bright range of new functionalities, like animations, geo-location and canvas (drawing). Also new UI elements for HTML forms are now available which support for example Smartphone specific on-screen keyboards [1]. HTML is a “living standard”, which means that it will continuously be improved and extended [2].

Identification of Capabilities

HTML5 is relatively new and no browser supports all of its features so far. However, most used browser like Firefox, Internet Explorer, Safari and Opera already implement major parts of it, and will likely support the whole standard rather soon. The major advantage of HTML5 is the high availability on all devices as long as they run an up-to-date browser. Therefore HTML5 is usable on all platforms as long as the browser supports it. HTML5 web sites can be created either manually or generated on-the-fly using PHP or Java Servlets, for example. Thus, HTML5 can be used in very flexible way [3].

HTML5 has been predicted to become more and more important in the near future [4].

Available on platforms:

Operating System	**Available**
Windows	Y
Linux	Y
OSX	Y
Android	Y
Symbian	Y
iOS	Y
BlackBerry OS	Y
Windows Phone	Y

Specific References

[1] 2011. Google Inc.*HTML5 Presentation*.[Online]. Available: http://slides.html5rocks.com/

[2] 2011. Hickson, Ian.*HTML5: Living Standard.* [Online]. Available: http://www.whatwg.org/specs/web-apps/current-work/multipage/

[2] 2011. unknown.*HTML 5 Demos and Examples*.[Online]. Available: http://html5demos.com/

[4] 2011. Taylor, Colleen.*HTML5 boom is coming, Fast*. [Online]. Available: http://gigaom.com/2011/07/22/the-html5-boom-is-coming-fast/

(B.4) Adobe Flash and Flex

An often used web technology is Adobe Flash which currently is used on a major number of web sites. Because of that it is more relevant compared to Java applets and Silverlight. Adobe Flash is majorly used for web site animations but also for building whole applications like video players. In contrast, Adobe Flex is made for business-like applications, which means mainly form-based applications. However, Flash and Flex applications will be compiled to the same kind of files (SWT-files) which means that both technologies can be treated equally in the context of platform support. [1][2]

Identification of Capabilities

For long, apart from Java applets Flash was the only possibility to develop real web applications which resulted in bright tool support and a fully developed standard which is currently the main advantage of Flash. Microsoft tried the same with Silverlight but Flash is still the better estab-

lished and supported standard. Currently, Flash might be a good choice for UI development but, it has been prognosticated that Flash will be replaced by HTML5 medium-term to long-term [3, 4]. An additional indicator supporting this is that Adobe is developing a HTML5-Editor [5].

Adobe Flash and Flex is already well supported by all major desktop operating systems and also by the majority of mobile operating systems. However, a disadvantage of Flash and Flex is that a plug-in is required for executing Flash in a browser and for some browsers is no plug-in available. Therefore, Flash is not available especially on Apple's iOS and Windows Phone 7.

Available on platforms [6]:

Operating System	**Available**
Windows	Y
Linux	Y
OSX	Y
Android	Y
Symbian	Y
iOS	**N**
BlackBerry OS	Y
Windows Phone	**N**

Specific References

[1] 2011. Adobe Systems Incorporated. *Adobe Flash Platform*.[Online]. Available: http://www.adobe.com/flashplatform/?promoid=ITXQR

[2] 2011. Adobe Systems Incorporated. *What is Flex?*.[Online]. Available: http://www.adobe.com/products/flex/

[3] 2011. Lee-Delisle, Seb. *HTML5 vs Flash – the aftermath*.[Online]. Available: http://sebleedelisle.com/2011/01/html5-vs-flash-the-aftermath/

[4] 2011. Hedemann, Falk. *HTML5 Boom: Wachstum verdrängt Flash (und spielt Apple in die Karten)*. [Online].Available: http://t3n.de/news/html5-boom-wachstum-verdrangt-flash-und-spielt-apple-323447/

[5] 2011. Brintrup, Saskia. *Flash-Konkurrenz aus den eigenen Reihen: HTML5-Design-Tool Edge startet als Preview.* [Online]. Available: http://www.basicthinking.de/blog/2011/08/01/flash-konkurrenz-aus-den-eigenen-reihen-html5-design-tool-edge-startet-als-preview/

[6] 2011. Adobe Systems Incorporated. *Flash Player Features*.[Online]. Available: http://www.adobe.com/products/flashplayer/features/

8.2. Template for Generic Enabler Requests

To document, collect, prioritize and trace the "backlog entries", i.e., the Generic Enabler Requests, the FI APP Architecture Board has agreed on a structured template. The fields of this template are listed below:

- Id: Used to unequivocally identify the backlog entry
- Name: Descriptive, short name of the entry
- Goal/Rationale: Short phrase describing the goal / rationale for this entry
- Version Number: Version associated to the entry. Helpful to monitor progress and follow-up modifications
- Source: Project or organization who identified the feature
- Source contact: Contact point in Source project or organization
- Stakeholder: List of additional projects or organizations interested in coverage of the feature (the source is considered to be a stakeholder)
- Scope:
    - *Platform* = functional or non-functional feature required at platform level
        - *Platform Generic* = It relates to a feature required at platform level and which has a general purpose
        - *Platform Common* = It relates to a functional or non-functional feature required at platform level but whose applicability is restricted to applications in a few number of domains (Usage Areas)
    - *Application* = functional or non-functional feature required at application level
    - *Global* = functional or non-functional feature required both at platform (generic or common) and application level
    - *Not Yet Determined*
- Status:
    - *Pending* = not yet developed / revised
    - *Planned* = is in the roadmap (in the "product backlog")
    - *Under execution* = being developed in current sprint
    - *Done* = has been developed
    - *Deprecated,* in this case the Description field should explain why and should state the list of Ids of entries replacing it
- MoSCoW priority:
    - *MUST* = feature that absolutely has to be developed; if any such feature were not developed, the project will be considered a failure.
    - *SHOULD* = features important to the success of the project, no absolute musts (there might be a workaround or the lack of that feature will not cause the project to fail)
    - *COULD* = features that is nice to have but is not a core features
    - *WONT* = feature that will not be implemented (as part of the project).
- Relative priority Number: Priority number relative to the same MoSCoW priority

- Chapter: Will refer to chapters in FI-WARE, when dealing with entries related to FI-WARE Generic Enablers:

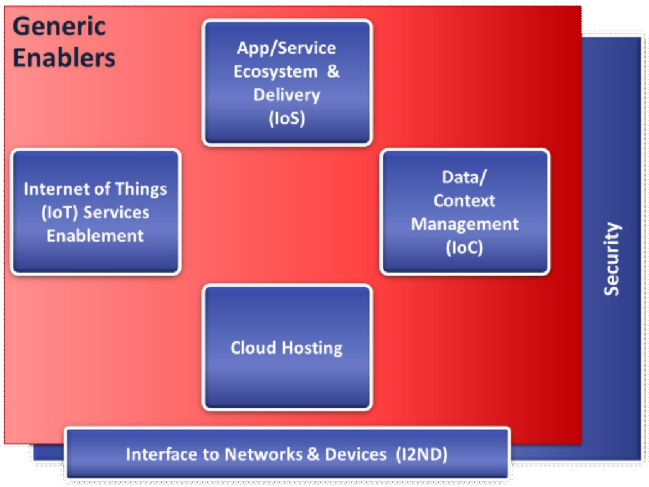

Figure 9: Overview of FI-WARE Chapters

- Enabler: Only applicable features with scope "Platform". It identifies the Generic Enabler to which this entry (feature) in the backlog applies.
- Category: Describes the granularity of the entry. Following Scrum this could be *Theme*, *Epic* or user *story* (i.e., feature)
- Description: The following is recommended to be provided but not mandatory:
  - *Actors*
  - *Primary Actors*
  - *Facades*
  - *Preconditions*
  - *Triggers*
  - *Main success scenario ("How to demo?")*
  - *Extensions*
  - *Alternative paths*
  - *Post-conditions*
  - *Notes*
- Owner: Name of project taking care of it
- Owner contact: Name of person in Owner project taking care of it
- Complexity: Description of how complex supporting the feature will be:
  - *XXL* = Costs quite a lot
  - *XL* = Costs a lot
  - *L* = Has a significant cost
  - *M* = Medium cost
  - *S* = Doesn't take that much
  - *XS* = It's almost trivial
- Creation Date: Date of creation
- Last modified: Last date at which the entry (any field) was modified