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

D7.3

Initial Technical Specification of the Transport Planning Component

Project Acronym	Finest	
Project Title	Future Internet enabled optimisation of transport and logistics networks	
Project Number	285598	
Workpackage	WP7 Transport Planning and Re-planning	
Lead Beneficiary	MARINTEK	
Editors	Marianne Hagaseth	MARINTEK
	Åsmund Tjora	MARINTEK
Contributors	Kay Endre Fjørtoft	MARINTEK
	Lone Ramstad	MARINTEK
	Christian Steinebach	MARINTEK
Reviewers	Metin Turkay	KOC
	René Fleischhauer	SAP
Dissemination Level	PU	
Contractual Delivery Date	30-09-2012	
Actual Delivery Date	30-09-2012	
Version	1	

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 is the third deliverable from FInest work package 7, "Transport Planning and Replanning", and contains updated requirements and the initial technical specification for a Future Internet Transport Planning Module that shall work together with other modules on the FInest platform.

The module shall support the generation of a transport chain plan consisting of several transport execution plans, where each transport execution plan documents the agreement between a logistics service client and a logistics service provider. The building of the plan is based on the transport demand that the planner describes in the module and the contract details and transport service descriptions retrieved from the E-Contracting Module and the logistics service providers' systems. The module shall support the use of the demand, contract and service information in order to generate transport execution plans, control that the individual execution plans match each other in a way that makes the total transport chain executable, and support the negotiation and booking process between the client and provider.

The technical specification documented in this deliverable consists of a component model, an interface model and an information model. The component model describes the components that the Transport Planning Module consists of and the intended functionality of each component. The interface model documents the interfaces and message data types of the Transport Planning Module that are used for communication with other FInest modules, the user front-end and the service provider systems. The information model describes the main information elements used in the Transport Planning Module as well as the relations between the elements.

The specification will serve as a base for prototype implementation and further design refinements in the last quarter of the FInest project.

Document History

Version	Date	Comments
V0.1	27-03-2012	Added technical work from M7-M12
V0.2	07-09-2012	Inserted results from technical specification work. Cleanup of document.
V0.3	13-09-2012	Updated design based on feedback from technical coordination.
V0.4	19-09-2012	Internal review version
V0.5	24-09-2012	Updates based on internal reviews
V1.0	27-09-2012	Final version

Table of Contents

1. Introduction 8

1.1. Brief description of the Transport Planning Module 8

1.2. Previous work on the Transport Planning Module 10

1.3. Relation to other work packages in the FInest project 11

1.4. Structure of this deliverable 11

2. Requirements and conceptual design updates 12

2.1. Additional module functionality 12

2.1.1. Building Transport Services 12

2.1.2. Maintenance of TSD information 13

2.2. Requirements handled outside the Transport Planning Module 13

2.2.1. Functionality handled by applications utilizing the module 14

2.2.2. Functionality not prioritized during FInest project 14

3. Initial Technical Architecture 14

3.1. Transport Planning Module component model 14

3.1.1. Demand and Service Builder 17

3.1.2. Transport Service Search 18

3.1.3. Transport plan generation and maintenance 19

3.1.4. Supporting components 22

3.1.5. Front-End description 23

3.1.6. Back-End 24

3.2. Transport Planning Module Interfaces 24

3.2.1. ComplianceChecks, MarketplaceInteractions, OnlineContractInformation 26

3.2.2. GetPlanningRules 26
3.2.3. LSPSystemsInterfaces 26
3.2.4. OptimizerInterfaces 27
3.2.5. Planning 27
3.2.6. ServiceSelection 28
3.2.7. TSDDescription 29
3.3. Transport Planning Module Data Types 30
3.3.1. TSDVariant 31
3.3.2. PriorityType 32
3.3.3. ReplanningType 32
3.3.4. RulesType 33
3.4. Module interactions 33
3.5. TPM Information Model 34
3.5.1. Overall Data Model 35
3.5.2. Transport Chain Plan 37
3.6. Use of Generic Enablers 37
4. Summary and future work 38
References 38

List of Figures

Figure 1 – Possible workflow for planning with the TPM 10
Figure 2 – TPM Conceptual Architecture from D7.2 12
Figure 3 – TPM Component model 16
Figure 4 – TPM Interfaces 25

Figure 5 – TPM Data types 31

Figure 6 – Sequence diagram for interactions between the TPM and other modules 34

Figure 7 – Initial TPM Overall Data Model 35

Figure 8 – TCP instantiated from GII 37

List of Tables

Table 1 - GetPlanningRules interface 26

Table 2 - LSPSystemsInterfaces interface 26

Table 3 - OptimizerInterfaces interface 27

Table 4 - Planning interface 28

Table 5 - ServiceSelection interface 28

Table 6 - TSDDescription interface 29

Table 7 - TSDVariant 32

Table 8 - PriorityType 32

Table 9 - ReplanningType 32

Table 10 - Initial TPM Information Model Description 35

Acronyms

Acronym	Explanation
BCM	Business Collaboration Module
CRUD	Create, Read, Update, Delete (basic operations on data e.g. in a database)
ECM	E-Contracting Module
EPM	Event Processing Module
GE	Generic Enabler
GII	Goods Item Itinerary
LSC	Logistic Service Client
LSP	Logistic Service Provider
TBD	To Be Defined
TCP	Transport Chain Plan
TEP	Transport Execution Plan
TP	Transport Plans
TPM	Transport Planning Module
TSD	Transport Service Description
UBL	Universal Business Language containing a library of standard electronic XML business documents such as purchase orders and invoices. It is developed by OASIS.

1. Introduction

This deliverable contains the main results in the work with the FInest Transport Planning Module (TPM) between months 13 to 18 of the FInest project. The purpose of the work is to design a module that can be used to help with the planning and replanning of a transport chain, by utilizing new possibilities that Future Internet technologies offer; the intended functionality includes describing the transport demand, finding and configuring transport services that fulfil the demand, and booking of the selected services. The module is designed to interact with other modules, including two of the three other modules developed in the FInest project, the E-Contracting Module (ECM) and the Business Collaboration Module (BCM). The module will also provide data that is used by the Event Processing Module (EPM).

The deliverable is the third deliverable from Transport Planning and Replanning work package (WP7), and documents the initial technical specification for the FInest Transport Planning Module. The technical design is based on the concepts and requirements that appear in the previous deliverables from the work package, *D7.1 - Requirements Analysis and Selection of Technology Baseline for Transport Planning Component* [1] and *D7.2 – Conceptual Design of the Transport Planning Component* [2].

The main contents of the initial technical specification are the refined component model that describes the components the module consists of, the intended cooperation with other FInest modules and intended use of FIWare Generic Enablers, the interface model that describes the interfaces of the module, and a data model describing the structure of the TPM database.

The document focuses on what is considered the main functionality, interface and structure of the TPM, but there is also some discussion of possible "nice-to-have" features that are considered out of scope for the FInest project, but that may be pursued in future versions of a Future Internet transport planner. The rationale for this is to illustrate future possible expansions to the TPM and some of the possibilities for cooperating modules outside the FInest project.

1.1. Brief description of the Transport Planning Module

The Transport Planning Module is a Future Internet module offering services for planning a transport chain, including finding transport offers that match the demand, negotiation of terms, bookings etc. Its functionality is roughly divided in three parts:

First, the module offers functionality for describing transport demands as well as transport services, so that they can be used in the later planning process. This also includes functionality to publish transport demands and services on other systems, e.g. marketplaces and information sites.

Second, the module offers functionality to keep track of transport services matching a demand. In a standard FInest setup, the module will cooperate with the ECM in order to find services from long-term contracts and spot market that can be used to fulfil the demand.

Third, the module offers functionality to generate and configure a Transport Chain Plan consisting of several Transport Execution Plans, based on the service and demand descriptions, as well as the functionality for negotiation on the services between the logistics service provider (LSP) and the logistics service client (LSC) and booking of the services. When a plan changes status from "under development" to "ready for execution", it will be transferred to the BCM.

The main outputs of the TPM are the Transport Chain Plan (TCP) describing the whole planned transport chain and the Transport Execution Plans (TEP) containing details on the individual parts of the chain. There will also be options for using the TPM for publishing transport demands (in the form of Transport Service Description Requests (TSD-request)) and Transport Service Descriptions (TSD) on external systems. The messages used will conform to the Common Framework model [3]. The Common Framework messages are also used in other EU transport projects, e.g. e-Freight [4], FreightWise [5], i-Cargo [6], DiSCwise [7] and others.

Figure 1 shows a possible workflow for planning with the TPM, based on the mock-up described in deliverable D7.2 [2], but the TPM will be designed so that other workflows and other ways of using the module are possible.

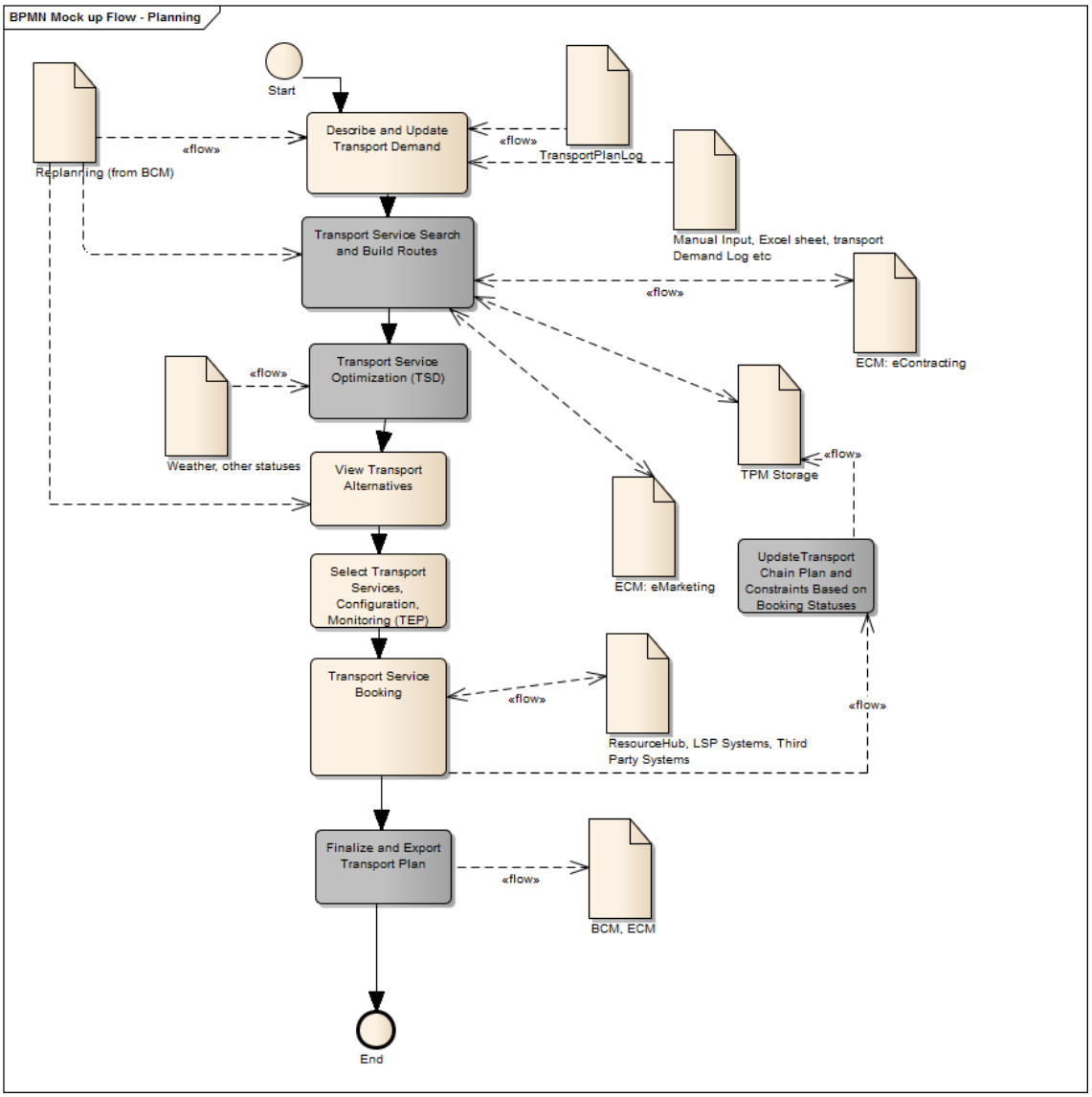

Figure 1 – Possible workflow for planning with the TPM

1.2. Previous work on the Transport Planning Module

Earlier work on the TPM has been documented in the deliverables D7.1 and D7.2.

Deliverable *D7.1 – Requirements Analysis and Selection of Technology Baseline for Transport Planning Component* [1] discusses different types of transport and logistics planning and the state of the art in the transport planning field. The discussions are used as a base for generating the initial analysis of the requirements to the TPM. The scope of the TPM is set to the creation of an overall operational transport chain plan for a multimodal transport chain by utilizing relevant and recent information at planning time.

Deliverable *D7.2 – Conceptual Design of the Transport Planning Component* [2] refines the requirements from D7.1 and describes functionality of the TPM by a set of Use Cases. The

report describes how the TPM can contribute to the collaboration platform to achieve more efficient planning processes and increased opportunities for utilizing and combining resources and service capabilities with transport demand descriptions. The main services that must be offered by the TPM are related to describing the transport demand, finding transport services matching described demands, and booking of selected transport services. In addition, services for configuring the transport services, both through automatic and manual means, as well as use of relevant information from the other FInest modules (e.g. contracts from the ECM) during transport planning has been described.

The scope of the TPM and the requirements described in D7.1 as well as the conceptual design, refinements and descriptions of functionality from D7.2 serve as the base for the technical specifications presented in this report.

1.3. Relation to other work packages in the FInest project

The work packages in the FInest project have been tightly coupled, both in order to identify realistic requirements and challenges from the transport and logistics domain, and to ensure that the different parts of the technical work are aligned with each other.

Work package 1, Domain Characterization and Requirements Analysis, is concerned with the identification of requirements from the transport and logistics domain. The identified requirements are used as contributions to the design of the TPM.

Work package 2, Use Case Specification, provides use case scenarios that serve as refinement of requirements, presentation of domain challenges, as well as experiment specifications for domain trials that utilizes the project's technical solutions.

Work package 3, Solution Design and Technical Architecture, provides the overall design and architecture for the FInest platform that the TPM shall be integrated into.

Work package 4, Experimentation Environment, is concerned with the development of an experimentation infrastructure for the use case trials defined in WP2.

Work packages 5, 6 and 8, Business Network Collaboration, Proactive Event Driven Monitoring and Logistics Contract Establishment and Management, are concerned with the development of other modules in the FInest platform. It is the intention to create designs that allow all the FInest modules cooperate closely in order to utilize each other's functionality.

Work package 9 is responsible for the alignment of the FInest project with other projects of the FI-PPP program.

1.4. Structure of this deliverable

The rest of the deliverable is structured as follows:

Chapter 2 contains updates to the requirements and design described in the D7.1 and D7.2 deliverables.

Chapter 3 describes the initial technical specification of the Transport Planning Module.

Chapter 4 contains a summary and a description of future work.

2. Requirements and conceptual design updates

The main requirements studies and the conceptual design for the Transport Planning Module have been reported in the D7.1 [1] and updated in the D7.2 [2] deliverables. D7.2 also described a set of use cases to show intended functionality of the TPM. This chapter contains updates to the requirements and design, including a brief discussion on functionality not discussed in earlier deliverables.

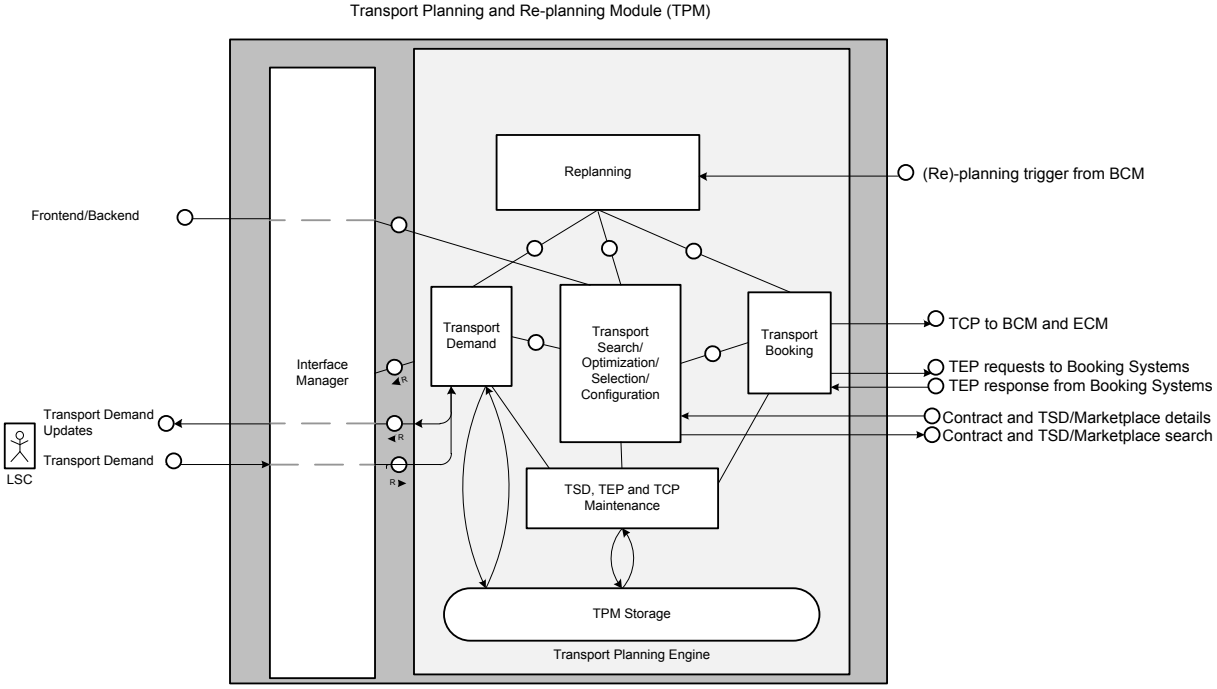

Figure 2 – TPM Conceptual Architecture from D7.2

Unless noted otherwise in this chapter, the updated requirements lists in D7.2 shall be used.

2.1. Additional module functionality

During the work with the technical specification, new functionality has been added to the design.

2.1.1. Building Transport Services

Earlier designs have functionality for describing the demands of an LSC. The functionality could be expanded to also offer description of transport services. Technically, the TSD-request used for demand description and the TSD used for service description are similar, and the TPM already have planned data structures for storage of service descriptions.

This will increase usability for LSPs that are planning a part of chain for their clients, as the tool also can be used to describe the services they offer.

In addition to tools for building service descriptions, the connection of FInest modules to external marketplaces must be utilized so that services built using the TPM are available on these marketplaces, and thus be visible to potential clients.

In order to make the TPM more useable to the LSP side, the TEP booking and negotiation functionality should be expanded to support both sides of the negotiation.

A TPM transport service builder could be connected to external tools for resource management, weather and traffic information and route planning, in order to update TSDs with real-time information on e.g. schedules and capacity. For the current design, it is assumed that external tools will use the standard interfaces for TSD updates, and these kinds of tools will not be investigated further in this deliverable.

2.1.2. Maintenance of TSD information

During the planning process, the module must be able to update the information in the TSDs that it has already found. Limited capacity or transport resource availability, changes in schedules etc. may result in changes in the service descriptions while the planning is in progress, and updates will be necessary when doing the planning. The following TSD update strategies should be supported:

- Polling: With this strategy, the TPM checks the LSP system for updates. The time between the checks may be dependent on the type of service, the remaining capacity on the service, and the time to plan finalization.
- Subscription: This strategy depends on the LSP system to publish to subscribed parties that the TSD has been updated.

If an update results in a TSD being unsuitable for the demand, it will be marked as unsuitable or removed from the list of usable services (depending on the user's preferences).

Note that this functionality is only necessary before generating the Transport Execution Plans. When a TEP negotiation starts, it is assumed that any changes will be made known during the negotiation process. When a TEP is finalized (i.e. agreement on the logistics service has been reached between the client and provider), it is assumed that changes to services will be handled by the BCM, possibly by requesting a replanning in the TPM.

2.2. Requirements handled outside the Transport Planning Module

It is worth noting that several of the requirements and functions from the TPM use cases described in D7.2 are not handled directly by the TPM itself. For many of these, delegation to the FInest platform or to external systems have already been noted in the D7.2 deliverable.

2.2.1. Functionality handled by applications utilizing the module

Some of the requirements are not related to the services that the planning module itself should offer, but that are useful in a planning environment where the module takes part.

The collaboration functionality, described in the TPM use case UC-TPM06, is considered to be on the application level, and thus not handled in the TPM. Several Internet services for collaboration and teleconferencing already exist, and these could be utilized by the application in order to achieve the described functionality.

2.2.2. Functionality not prioritized during FInest project

Some functionality in the studies from D7.2 is considered useful for a transport planning component, but may not be necessary for the basic functionality of the Transport Planning Component while being fairly complex to design properly. Such functionality may get lower priority during the detailed technical design, and may be mentioned as future expansions of the TPM or as possible services that can cooperate with the TPM. If such functionality is omitted in the detailed design, the design must still be open for the insertion of the functionality at a later stage.

The optimization routines mentioned in TPM use case UC-TPM02-B can be quite complex and computation-heavy, and while good optimization is very useful in the planning process, the design of such a tool will not be prioritized during the project. It is also likely that such a tool, when created, can be designed as an add-on module accessing the interfaces of the TPM, as the module also must be designed to support external optimizer tools.

3. Initial Technical Architecture

This chapter contains the initial technical architecture of the TPM. The main parts of the architecture discussed here are the component model in section 3.1 that describes the component structure of the module, the interface model in section 3.2 and data types in section 3.3 describing the interfaces and messages used when communicating with the module, and the information model in section 3.5 describing the data structure used by the TPM. The chapter also contains short descriptions of interaction with other FInest modules in section 3.4 and a discussion on the use of Generic Enablers in section 3.6.

The conceptual architecture described in D7.2 has been used as a base for the work. This has been restructured and refined in order to generate the technical architecture. Another important part of the work has been to coordinate design of module interfaces, interactions and responsibilities with the other technical work packages, in order to align with the technical specification deliverables from these work packages [8] [9] [10] [11].

3.1. Transport Planning Module component model

The TPM consists of three main components: The **Demand and Service Builder** component is used creating and updating transport demands and services that are used in the planning process, the **Service Search** component is used for searching for and updating transport services that can

be used for fulfilling the demand, and the **Transport Plan Generation and Management** component is used for generating the actual transport plans, configuring the plans, as well as the negotiation and booking for the transport services. The component model is shown in Figure 3.

The TPM architecture follows the Model-View-Controller (MVC) paradigm [12] characterized by:

- Model: Keep the knowledge of the system, for instance the statuses and states. This is the TPM itself
- View: The presentation of the model. This is the available widgets or apps in the front-end/collaboration space that is used to interact with the model, that is, with the TPM.
- Controller: This is the link between the user (the view) and the system (the TPM). The controller listens to updates in the view and changes the state in the model according to this. The controller makes the model and view independent of each other, and each of them can be reusable. However, the Controller is not easily reusable since it is dependent on both the model and the view.

The Collaboration Space/Front End constitutes the basic building blocks (apps or widgets) that are available to tailor the use of the TPM (and the use of the rest of the FInest) to the needs of each specific user.

Examples of apps or widgets that may be supported by the services offered in the TPM:

- Editing and handling Demand Descriptions
- Editing and handling Service Descriptions
- Service Selection and Configuration
- Service Negotiation and Booking
- View Plans.

The views offered by the TPM can be combined with views from other modules and providers, for instance apps for collaboration (for instance teleconferencing services), apps describing the weather along the route of the planned transport, or contract search from the ECM (the planner may decide whether he want to search among contracts or not).

A mash-up tool (or something similar) can be used to set up how the various views can be adapted into a working environment for the end user. Examples of rules that should be possible to set up in the mash-up as seen from the TPM, are:

- Decide whether to do contract search or not
- Decide whether to do service search or not
- Restrict the set of LSPs that we want to search among

The TPM (model) contains the components needed to realize the TPM functionality including the storage of TSDs, TEPs and TCPs.

The backend describes interaction with other FInest core modules and LSP legacy systems.

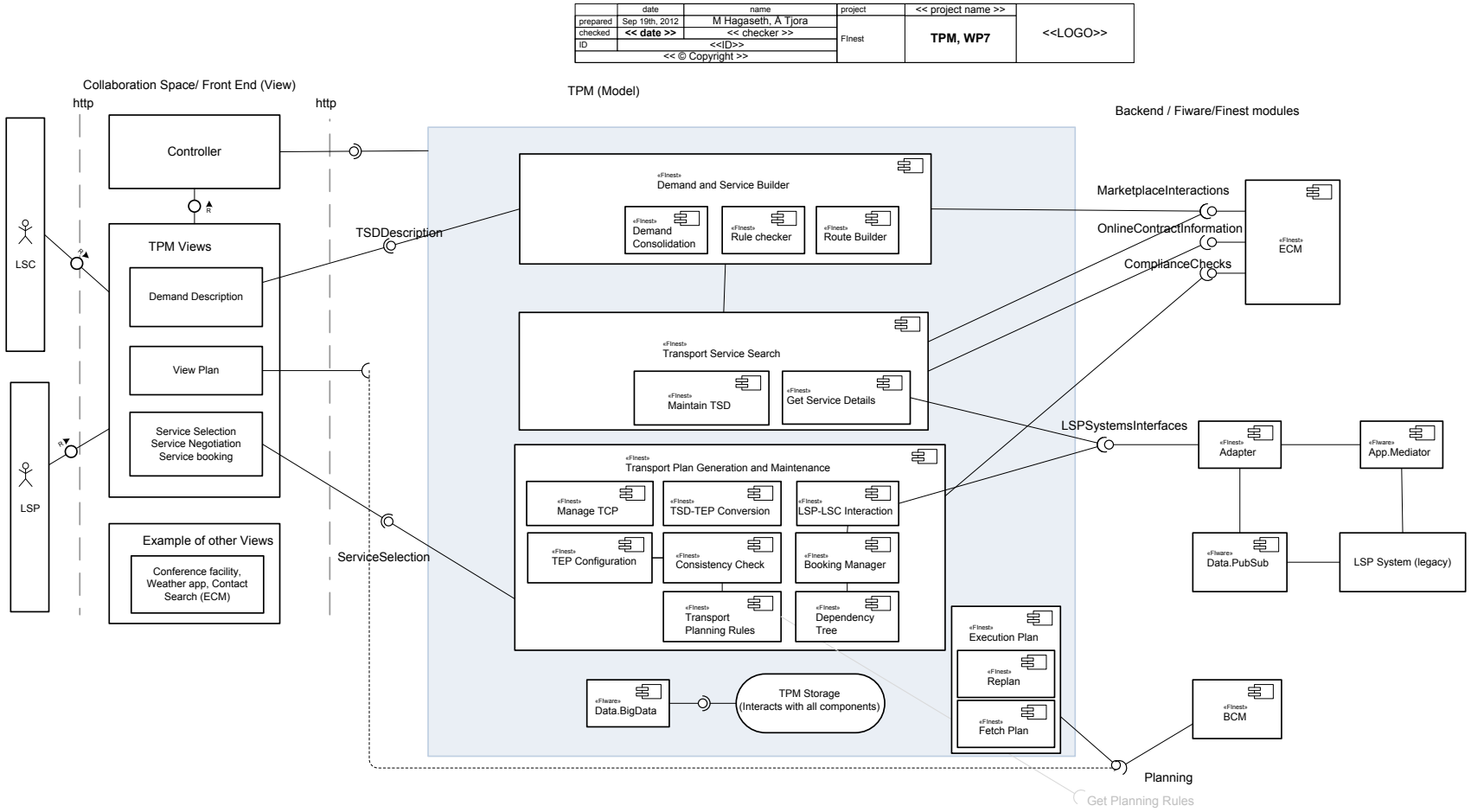

Figure 3 – TPM Component model

3.1.1. Demand and Service Builder

The main purpose of this component is to build valid Transport Service Descriptions (TSD) in the TPM Storage. The TSD structure can be used both for describing an offered transport service and a demand for transport; the component must be able to handle building of TSDs for both purposes.

For this reason, CRUD operations on TSD will be the central part of this component, and the main interface to the front-end will be for accessing these CRUD operations.

In addition to the building of the TSDs in the internal database, the component shall also have functionality to publish demands and services to external systems, e.g. marketplaces and information hubs.

Note that the focus of the TPM has been planning from the Logistic Service Client's viewpoint, thus most of the added functionality and subcomponents in this component will be for supporting the *demand* building.

Summary of functionality:

- CRUD operations on TSD, including TSD-request
- Publish TSD and TSD-request
- Building of routes for the transport (delegated to "route builder" subcomponent)
- Check that a TSD or TSD-request is ready for planning or publication, and generate feedback on missing parts (delegated to "rule checker" subcomponent)
- Consolidate demands from a set of TSDs (delegated to "demand consolidation")

3.1.1.1. Route builder

This component offers functionality for describing the route that the goods will take for fulfilling a demand. The TPM does not offer any automatic route building, but the interfaces for this subcomponent should be accessible both for manual route building and automatic route building tools.

Summary of functionality:

- Describe routes of a transport, by using location information for transport leg endpoints.
- Create multiple TSD-requests based on the full demand description and the separate parts of the route.

3.1.1.2. Rule checker

This component handles rule checking for building valid service and demand descriptions, and detects lack of required information. The criterion is the minimum information to be able to start planning with the TSD.

Summary of functionality:

- Check that a TSD-request fulfils a minimum set of information.
- Check that a TSD fulfils a minimum set of information.

- Check if additional information is needed, based on the information already entered (e.g. if goods in the demand is classified as hazardous materials)
- Generate feedback on what kind of information is needed in order to get a valid TSD.

3.1.1.3. Demand consolidation

The component must support consolidation of demands, and could also add functionality for finding demands with similarities with respect to time period and locations for an overview of demands that could be consolidated in a simple manner. The component will not offer advanced consolidation tools, but may offer interfaces for external tools used for this purpose.

Summary of functionality:

- Search for open demands handled by the same user that can be consolidated with the current demand, based on similar location and timing.
- Modify TSDs to reflect consolidation, including creation of a consolidated demand description and adding consolidation and deconsolidation steps into the requested chain.

3.1.2. Transport Service Search

The purpose of this component is to initiate the search for services that match a demand, and store and maintain the service descriptions, so that they are updated.

The actual search for services are done by external modules (in FInest, the ECM has this functionality).

A service search may be done in three different ways:

- **Search for contracts:** The planner may have long-term contracts with LSPs that can fulfil (parts of) the demand. If a matching contract is found, the search for relevant TSDs will be done at the contract partner's systems.
- **Search for services on spot market:** The search for relevant TSDs will be done at the spot market.
- **Use predefined TSDs:** The search will be done on services already stored in the system. This is relevant when e.g. service providers generate plans where their own services are part of the transport chain.

The component should be designed so that future versions of the component also can support service providers matching demands to their services, thus making the module a better tool for the LSP.

Summary of functionality:

- Communication with ECM, marketplaces, and LSPs in order to get transport service descriptions (delegated to "Get service details")
- Storage of a pool of TSDs that can be used for fulfilling the demand
- Maintenance on the pool of TSDs so that they are reasonably updated (delegated to "Maintain TSD")

3.1.2.1. Get service details

This subcomponent is responsible for getting TSDs that match the plan. This includes the contact with the ECM and marketplaces, as well as contact with LSP systems.

Summary of functionality:

- Request relevant contract data from contract management system (e.g. ECM)
- Request relevant TSD data from ECM and external sources
- Request subscription to updates of relevant TSDs
- Find relevant TSDs in user-specific storage

3.1.2.2. Maintain TSD

This subcomponent runs the rules for when updates to TSDs should be searched for, when TSDs should be automatically removed from the pool (e.g. when a description no longer matches the demand) etc.

Summary of functionality:

- Decide when a TSD must be updated, according to a set of rules.
- Mark TSDs or remove them from pool when they are no longer usable to fulfil the request.

3.1.3. Transport plan generation and maintenance

This component's main purpose is to generate and configure the Transport Execution Plans (TEP) that the Transport Chain Plan (TCP) consists of, and to facilitate negotiation and booking of TEPs.

The main steps for creating an executable transport plan will be:

- **Create initial TEP and TCP:** Convert demand, service and contract information into Transport Execution Plans, and build a Transport Chain Plan based on the TEPs.
- **Configure TEP:** This is any configuration to the TEP done after its initial creation.
- **Negotiate and book TEP:** This step includes any negotiation between the LSP and the LSC on the contents of the TEP. Booking is done when the contents are agreed upon.

Summary of functionality:

- Maintain a TCP describing the end-to-end transport chain for the demand (delegated to "Manage TCP")
- Convert TSDs describing services and demand as well as known contract information to an initial TEP (delegated to "TSD-TEP Conversion")
- Find and warn about any inconsistencies in the transport chain when TEPs are updated or deleted (delegated to "Consistency Checker")

- Configuration of TEPs (in the simplest form, this can be done by CRUD operations on the TEP, but a layer of logic and rules could be applied on top) (delegated to "TEP Configuration").
- Apply location-, goods- and mode-specific rules to the consistency check and TEP configuration (delegated to "Transport Planning Rules")
- Generate and advise on the best booking strategy in order to minimize the number of cancellations necessary in case of failed bookings (delegated to "Dependency Tree")
- Support the negotiation and booking of TEPs (delegated to "LSP-LSC Interaction")

3.1.3.1. TSD-TEP Conversion

This subcomponent is responsible for creating Transport Execution Plans based on the Transport Service Descriptions for the demand and the services, as well as information that can be extracted from contracts and user preferences.

Summary of functionality:

- Generate a new TEP.
- Add data from TSD-request (demand) to TEP
- Add data from TSD (service) to TEP
- Add contract information to TEP
- Place TEP reference in TCP (call to "Maintain TCP")

3.1.3.2. Manage TCP

This subcomponent is responsible for managing the Transport Chain Plan data.

Summary of functionality:

- Generate a new TCP (this may be done when a demand is created)
- Add TEP references when new TEPs are generated
- Remove TEP references when TEPs are deleted or cancelled.
- Keep track of parts of a transport chain that is unplanned (i.e. has no TEP); this includes keeping track of the demand descriptions for the unplanned part as well as any service descriptions that can be used to fulfil the parts.

3.1.3.3. Consistency Checker

This subcomponent is responsible for making sure that there are no *logical* inconsistencies in the plan, e.g. with respect to the timing between individual TEPs. The checker will be called when a TEP is changed. The results of the checks will be in the form of warnings that may be manually or automatically handled.

Summary of functionality:

- Check the TEPs that are part of a TCP with regard to a basic ruleset (e.g. timing, location).

- Check parts of a TCP with regard to location, goods and mode rulesets (e.g. required by laws). The determination of which rules that apply to any given chain is done by the "Transport Planning Rules" component.
- Generate warnings based on violations of the rules.

3.1.3.4. TEP Configuration

This subcomponent is used for any change to the TEP after its initial creation. Any parts of the TEP may be changed, but warnings will be given if the changes do not pass a consistency check. It should also be noted that the TEP are to be negotiated between the LSP and LSC; it is no guarantee that a change is acceptable by the negotiation partner.

Summary of functionality:

- Offer CRUD operations to TEPs.
- Call methods in "Consistency Checker" whenever updates are made.
- The component may also offer a layer of logic to ease the operations on the TEP configuration; this may however also be a part of the frontend functionality.

3.1.3.5. Transport Planning Rules

This subcomponent is used for keeping track of special planning rules related to locations, cargo type, mode etc. These are typically rules for timing and types of import and export clearance, special documentation handling needed in the chain etc. It is assumed that storage and maintenance of rulesets used are external to the TPM, and that this component fetches relevant rule sets from external sources.

Summary of functionality:

- Keep track of external sources containing rules that can be applied to transport
- Get the rules relevant to a given transport chain

3.1.3.6. Dependency Tree

This subcomponent will build a tree of the TEPs in a TCP, based on the dependency between the individual TEPs, and availability of TSDs that can be used to replace the selected services. This is done to find the best order of booking the different TEPs, with the aim of reducing the need for cancellations if the booking of one TEP should fail.

Summary of functionality:

- Generate a tree for booking order of the TEPs in a TCP.

3.1.3.7. LSP-LSC Interaction

This subcomponent is responsible for the contact between the provider and client in a TEP negotiation and booking process. The functionality should have support for using two-phase commit booking so that this can be used when it is also supported by the systems on the other side of the negotiation and booking process.

Summary of functionality:

- Transfer of TEPs between the parties in the negotiation process. Note that booking is considered the final step of the negotiation.
- Find differences between received and stored TEPs (for highlighting negotiation partner's changes).

3.1.3.8. Booking Manager

This subcomponent will manage the negotiation and booking process for whole TCPs.

Summary of functionality:

- Keep track of the negotiation and booking progress for the TEPs in a TCP, including keeping logs of the interactions.
- Management of automated booking processes, both standard and two-phase booking.

3.1.4. Supporting components

These components are parts of the TPM that are outside the tree main components.

3.1.4.1. TPM Storage

This is the main storage for the TPM, and is accessed by all components. The initial internal TPM database structure is discussed in section 3.5.

The database will utilize the Data.BigData Generic Enabler (see section 3.6).

3.1.4.2. Execution plan

This component handles tasks and interfaces related to several of the main components, and is mainly used for accessing the plans and the communication with the transport execution environment (i.e. BCM).

The two main subcomponents are the "Replan" and "Fetch plans" components.

3.1.4.3. Replan

This component is responsible for receiving replanning triggers, and invoking methods in the TPM based on the trigger contents. Typical actions will be:

- Update the demand to reflect the new situation.
- Initiate search for new services that can be used to replace the parts of the plan that failed.
- Maintenance on affected TCP and TEPs.

Note that the actual replanning is done with user interaction in the same way as the original planning; the TPM will not do the selection of services or booking of new plans automatically.

Summary of functionality:

- Receive the "Replanning Trigger" message.

- Generate updates to the demand description to reflect the situation.
- Initiate a new planning process based on the new demand
- Generate updates so that failed (and potentially failing) TEPs are handled.

3.1.4.4. Fetch Plan

This component is responsible for letting users and external systems or modules access the plans (i.e. through the Fetch Plan interface). The plans may be on any stage of planning, and may thus include unfulfilled demands, potential services that can be used to fulfil the demands, and TEPs in different stages of planning and booking. As the module will not handle the execution of the plan, the status on transport execution must be handled by other modules (i.e. BCM in the standard FInest setup).

Summary of functionality:

- Send signal to interested parties (e.g. BCM) when a plan is ready for execution.
- Send signal to interested parties (e.g. BCM) when a plan that has been marked "ready for execution" is updated.
- Provide interface for fetching TCPs, TEPs and TSDs.

3.1.5. Front-End description

This section contains the description of possible front-end views that can use the TPM functionality. The views will be realized by Widgets, small client-side applications that can be combined to suit the user's preferences. Widgets are further described in deliverable D3.3 [8]. It should be noted that the front-end is not a part of the TPM design; the views described here are examples that are presented with the intention to show how the technical interfaces may be used by planning applications.

Examples of possible user interface front-ends can also be found in the FInest mockup demonstrators, described in the *Conceptual Architecture* deliverables from the project, D5.2 [13], D6.2 [14], D7.2 [2] and D8.2 [15]; the mockup demonstrator described in D7.2 shows the use of FInest as a planning tool, but the other mockups may also show some functionality that can be realized with the TPM services.

The frontend may also contain non-FInest applications to ease the planning process; this may include collaboration tools (e.g. teleconferencing functionality), connection to tools for order management or resource management, or tools used for viewing information on the transport services that are outside the scope of the FInest modules.

Examples of views related to the TPM are:

- View Plans: This view displays a plan or a set of plans for a specific user, e.g. a LSC or a LSP. The TPM will handle requests of plans that are in the planning stage; for plans that have entered the execution phase, interaction with the BCM should be used for view of the transport status.
- Demand description: This view can include tools for the description of demands, and how the TPM shall perform the planning process with respect to the usage of contracts, usage of marketplaces and search criteria.

- Service description: This view can include the tools for the description of services, both for internal use (e.g. as a part of the user's own transport chain) and for publishing on marketplaces.
- Service selection and configuration: This view can include tools for selecting services to fulfil a demand, as well as configuring the services.
- Service negotiation and booking: This view can include interfaces for negotiation and booking of the services, in order to finalize the TEP.

3.1.6. Back-End

The module must also be able to communicate with LSP systems, both for getting access to transport service descriptions and for negotiating and booking of transport execution plans. If an LSP publishes updates to relevant TSDs, the TPM should be able to subscribe to these updates.

As the legacy systems used by the LSP are heterogeneous, the interfacing must be done via adapters. The design of adapters is considered to be out of the scope for the module itself.

3.2. Transport Planning Module Interfaces

The Transport Planning Module needs interfaces to the ECM for contract details, marketplace connections and compliancy checks, to the BCM for transferring the finished plans to execution and for receiving replanning triggers. The module must also have connections to the logistics service providers' systems for getting updated transport service descriptions and for the negotiation of plans and booking of transport. There is also need for connection to the frontend for user description of demand and services and user control of the planning process, including configuration of the plans.

This section describes the identified interfaces and interface methods. An overview is shown in Figure 4.

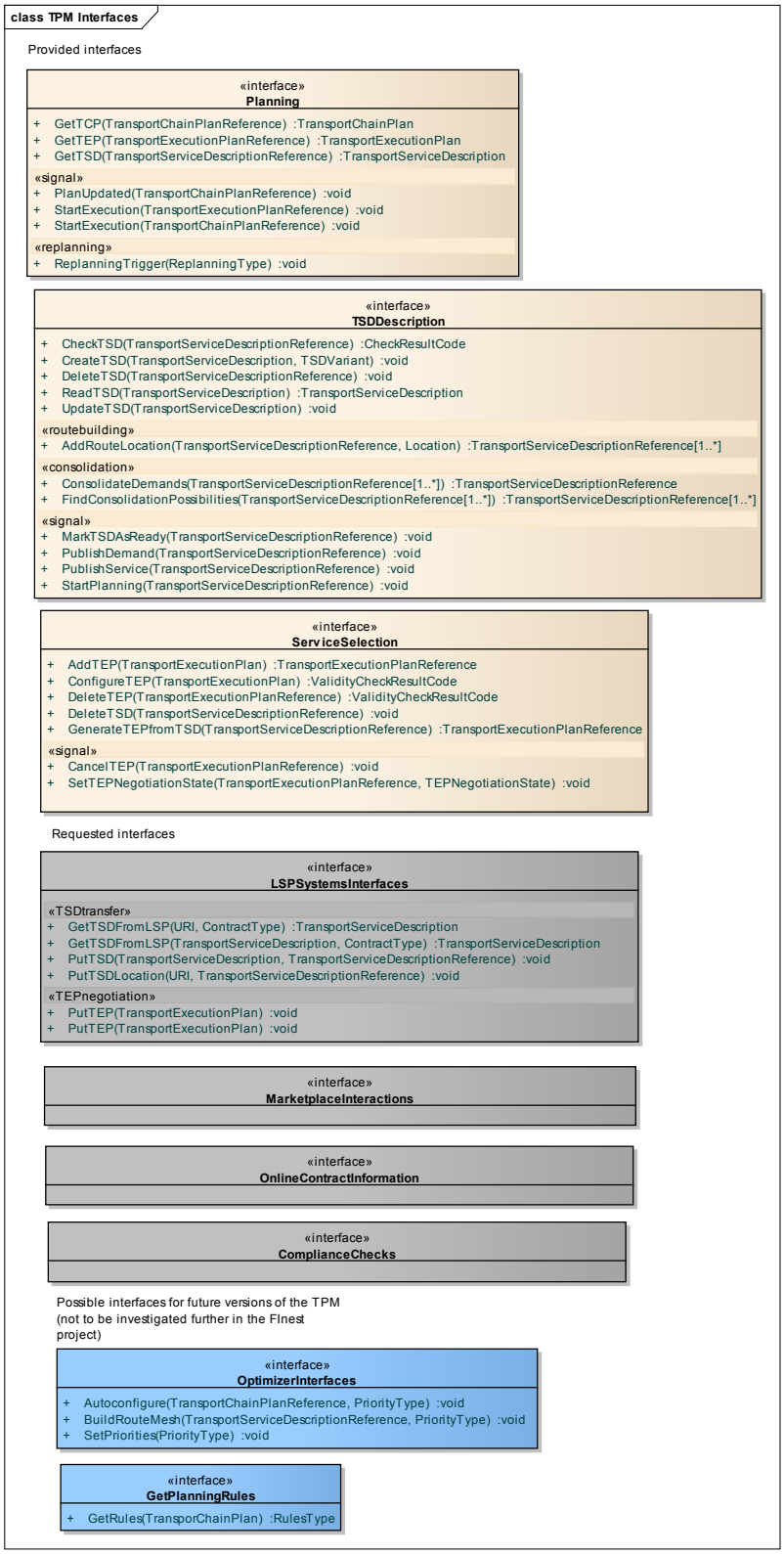

Figure 4 – TPM Interfaces

3.2.1. ComplianceChecks, MarketplaceInteractions, OnlineContractInformation

These interfaces are provided by the ECM, and are used by the TPM for checking the plans against contract information, interactions with marketplaces and for getting contract information. The initial interface descriptions are found in deliverable D8.3 [11].

3.2.2. GetPlanningRules

Proposed required interface for getting external rules when setting up a transport chain. Rules may be practical (e.g. goods must be at terminal at least 1 hour before loading) or legal (e.g. goods must wait at least 24h at customs storage).

This is an optional part of the TPM, and will not be fully designed or implemented in the prototype versions. The design should however be open for this option in future versions.

Table 1 - GetPlanningRules interface

Method	Notes	Parameters
GetRules() RulesType Public	Get rules affecting a Transport Chain Plan	**Transp… ChainPlan** [in] TCP

3.2.3. LSPSystemsInterfaces

Interface for contact with provider systems.

These are used for two main purposes; getting the TSD information from the LSP, and negotiation and booking of TEPs.

As external systems are widely heterogeneous, it is assumed that adapters are needed for the communication.

Table 2 - LSPSystemsInterfaces interface

Method	Notes	Parameters
GetTSDFromLSP() TransportServiceDescription Public	Requests an updated TSD directly from a known location in the LSP system	**URI** [in] TSDLocation **ContractType** [in] Contract
GetTSDFromLSP() TransportServiceDescription Public	Requests a TSD directly from an LSP based on a demand description	**TransportServiceDescription** [in] Demand **ContractType** [in] Contract
PutTEP() void Public	Sends a TEP *to* the TPM for negotiation (i.e. TPM is the receiver)	**TransportExecutionPlan** [in] TEP
PutTEP() void	Sends a TEP to LSP for negotiation (i.e.	**TransportExecutionPlan** [in] TEP

Method	Notes	Parameters
Public	TPM is the sender)	
PutTSD() void Public	Send a TSD to the TPM with a reference to the demand the TSD is expected to fulfil.	**TransportServiceDescription** [in] TSD **TransportServiceDescriptionReference** [in] DemandReference
PutTSDLocation() void Public	Send TSD's location to TPM, with a reference to the demand the TSD may fulfil. The TPM is responsible for getting the TSD from the location.	**URI** [in] TSDLocation **TransportServiceDescriptionReference** [in] DemandReference

3.2.4. OptimizerInterfaces

Possible interface to optimizing components.

Use of the methods here will initiate optimization and autoconfiguration of transport route or chain based on user-defined priority.

This is an optional part of the TPM, and will not be fully designed or implemented in the prototype versions. The design should however be open for this option in future versions.

Table 3 - OptimizerInterfaces interface

Method	Notes	Parameters
Autoconfigure() void Public	Initiates autoconfiguration of the parts of a transport chain plan	**TransportChainPlanReference** [in] TCPReference **PriorityType** [in] Priority
BuildRouteMesh() void Public	Initiates the automated building of a route mesh based on a transport service description	**TransportServiceDescriptionReference** [in] TSDReference **PriorityType** [in] Priority
SetPriorities() void Public	Sets the priorities to be used by optimization tools	**PriorityType** [in] Priority

3.2.5. Planning

Interface to the plan execution facilities and plan viewers.

The view plan methods will only return the parts known by TPM - for execution status, other components should be called.

Note that this will both be used by the frontend (e.g. letting the user view the plans) and for communication between TPM and BCM.

Table 4 - Planning interface

Method	Notes	Parameters
GetTCP() TransportChainPlan Public	Returns the Transport Chain Plan (containing TEPs and open TSDs) that is referred to.	**TransportChainPlanReference** [in] TCPReference
GetTEP() TransportExecutionPlan Public	Views a Transport Execution Plan	**TransportExecutionPlanReference** [in] TEPReference
GetTSD() TransportServiceDescription Public	Views a TSD. This may be both a demand and a description of an offered service.	**TransportServiceDescriptionReference** [in] TSDReference
PlanUpdated() void Public	Signals (e.g. to BCM) that a TCP that previously has been marked as "ready for execution" is updated.	**TransportChainPlanReference** [in] TCPReference
ReplanningTrigger() void Public	Sends a replanning request to the TPM. This may start updates to demand and search for new services in order to replace the failed services, but the actual replanning process will otherwise work as a normal planning process.	**ReplanningType** [in] ReplanningTrigger
StartExecution() void Public	Signals to the TPM that single transport execution plan is ready for execution.	**TransportExecutionPlanReference** [in] TEPReference
StartExecution() void Public	Signals to the TPM that a chain plan is ready for execution.	**TransportChainPlanReference** [in] TCPReference

3.2.6. ServiceSelection

Interface for selection of TSDs and creation and configuration of TEPs

The ValidityCheckResultCode is to be defined, but should contain values for "OK" (no problems detected), "Chain Problem" (the configured TEP does not fit with other parts of the transport chain), "No fit with transport description" (the configured TEP breaks rules set by the LSP in the TSD).

Table 5 - ServiceSelection interface

Method	Notes	Parameters
AddTEP() TransportExecutionPlan Reference Public	User adds a TEP (i.e. not generated by system)	**TransportExecutionPlan** [in] TEP
CancelTEP() void Public	Cancels a TEP that has entered the negotiation process or has been booked.	**TransportExecutionPlanReference** [in] TEPReference

Method	Notes	Parameters
	The TPM must contact the negotiation partner.	
ConfigureTEP() ValidityCheckResultCode Public	Updates a TEP. The system will perform a check whether the whole chain is still valid after the updates, and return the results of the check to the user.	**TransportExecutionPlan** [in] TEP
DeleteTEP() ValidityCheckResultCode Public	Deletes a TEP. The system will perform a check whether the whole chain is still valid after the updates, and return the results of the check to the user.	**TransportExecutionPlanReference** [in] TEPReference
DeleteTSD() void Public	User deletes a TSD. This will no longer be available for planning (but can be requested again from the source).	**TransportServiceDescriptionReference** [in] TSDReference
GenerateTEPfromTSD() TransportExecutionPlanReference Public	Generates a new TEP based on demand and service information.	**TransportServiceDescriptionReference** [in] TSDReference
SetTEPNegotiationState() void Public	Sets the negotiation state of the Transport Execution Plan	**TransportExecutionPlanReference** [in] TEPReference **TEPNegotiationState** [in] State

3.2.7. TSDDescription

Interface for performing operations on the Transport Service Descriptions.

The TSD may be either a demand (e.g. a TSDrequest) or a service (TSD).

Table 6 - TSDDescription interface

Method	Notes	Parameters
AddRouteLocation() TransportServiceDescriptionReference[1..*] Public	Adds a location in a route for a demand or service. This may generate several TSDs to cover different possibilities in a route mesh. References to these possibilities are returned.	**TransportServiceDescriptionReference** [in] TSDReference **Location** [in] Location
CheckTSD() CheckResultCode Public	Checks the completeness and validity of a TSD, e.g. whether the TSD can be used for planning as it is, or if more information must be submitted.	**TransportServiceDescriptionReference** [in] TSDreference
ConsolidateDemands() TransportServiceDescrip	Consolidates a set of demands.	**TransportServiceDescriptionReference[1..*]** [in] Demands

Method	Notes	Parameters
tionReference Public		
CreateTSD() void Public		**TransportServiceDescription** [in] TSD **TSDVariant** [in] ServiceOrDemand
DeleteTSD() void Public		**TransportServiceDescriptionReference** [in] TSDReference
FindConsolidationPossibilities() TransportServiceDescriptionReference[1..*] Public	Searches among the demands available to the user, and returns a set of demands that could potentially be consolidated.	**TransportServiceDescriptionReference[1..*]** [in] SearchCriteria
MarkTSDAsReady() void Public	Marks a TSD as ready to be used in planning processes for users with access to this TSD.	**TransportServiceDescriptionReference** [in] TSDReference
PublishDemand() void Public	Publishes a demand on a marketplace. Note that this does not start an active search for services.	**TransportServiceDescriptionReference** [in] TSDReference
PublishService() void Public	Publishes a service on a marketplace	**TransportServiceDescriptionReference** [in] TSDReference
ReadTSD() TransportServiceDescription Public		**TransportServiceDescription** [in] TSDReference
StartPlanning() void Public	Starts the planning process with the given demand	**TransportServiceDescriptionReference** [in] TSDReference
UpdateTSD() void Public		**TransportServiceDescription** [in] TSD

3.3. Transport Planning Module Data Types

The interfaces of the TPM will make use of the Transport Execution Plan (TEP) and Transport Service Description (TSD) messages described by the e-Freight framework [16], the messages are also part of UBL 2.1 [17]. The Transport Chain Plan (TCP) data type is based on the Goods Item Itinerary (GII) message also described in UBL 2.1. The TCP is further discussed in section 3.5.2.

In addition, the TPM will make use of the data types defined by deliverable D8.3 [11] for the interfaces with the ECM.

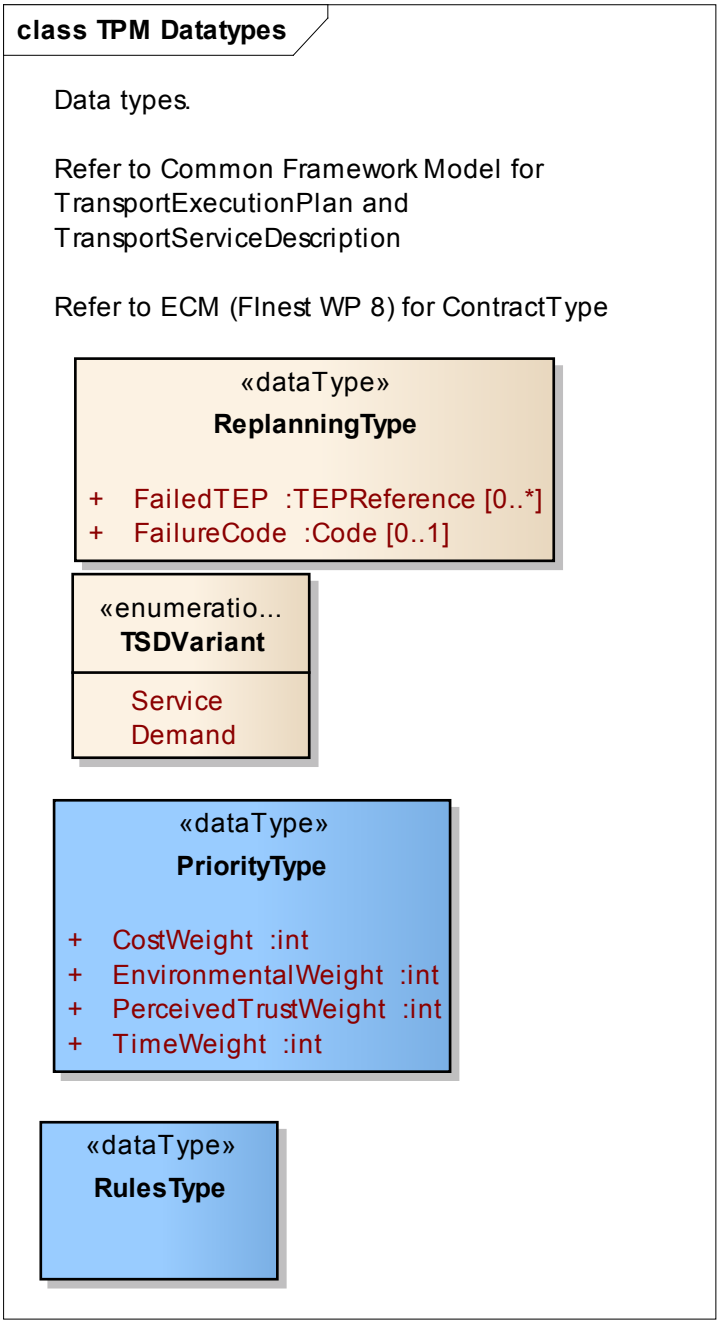

Figure 5 – TPM Data types

3.3.1. TSDVariant

Indication on whether a newly generated TSD describes a service or a demand.

Note that this may be expanded if necessary, e.g. to indicate whether a transport service runs in line or tramp traffic, or if it is agent service.

Table 7 - TSDVariant

Attribute	Notes	Constraints and tags
Service Public «enum»	Use this value to show that the created TSD describes a transport service.	*Default:*
Demand Public «enum»	Use this value to show that the created TSD describes a demand (i.e. is a TSDrequest).	*Default:*

3.3.2. PriorityType

This type is used for setting up priorities for automatic route finding and configuration of a transport chain.

As both route finding and external autoconfiguration is outside the scope of the TPM during the FInest project, this data type is just an example, and may be changed.

Table 8 - PriorityType

Attribute	Notes	Constraints and tags
CostWeight int Public	The weight optimizing routines should put on transport cost.	*Default:*
EnvironmentalWeight int Public	The weight optimizing routines should put on environmental issues (e.g. CO_2 emissions).	*Default:*
PerceivedTrustWeight int Public	The weight optimizing routines should put on using trusted service providers (with respect to ability to deliver the services as described).	*Default:*
TimeWeight int Public	The weight optimizing routines should put on the time of the transport.	*Default:*

3.3.3. ReplanningType

Data type for replanning status.

The intention is to give information needed to start replanning, e.g. where the original plan has failed, and what kind of failure has occurred.

Table 9 - ReplanningType

Attribute	Notes	Constraints and tags
FailedTEP TEPReference [0..*]	Refer to one or more TEPs that have (or potentially will) fail.	*Default:*

Attribute	Notes	Constraints and tags
Public		
FailureCode Code [0..1] Public	Gives a status code for the TEP failures. Coding TBD, but must cover situations like - Failure to pick up cargo - Failure to deliver cargo within deadline - Predicted failures (e.g. known delays, problems following failure in previous TEP etc.)	*Default:*

3.3.4. RulesType

Data type for rules that must be taken into consideration when planning.

As the GetPlanningRules interface is outside the scope of the TPM during the FInest project, this data type is just an example, and may be changed.

3.4. Module interactions

The module interacts with two other FInest modules, the ECM for contract information and marketplace connections, and the BCM for the execution of the plans. The plans generated by the TPM will also have some of the event processing parameters used by the EPM, but the BCM is responsible for the connection to this module. A sequence diagram showing the interaction between the modules is shown in Figure 6.

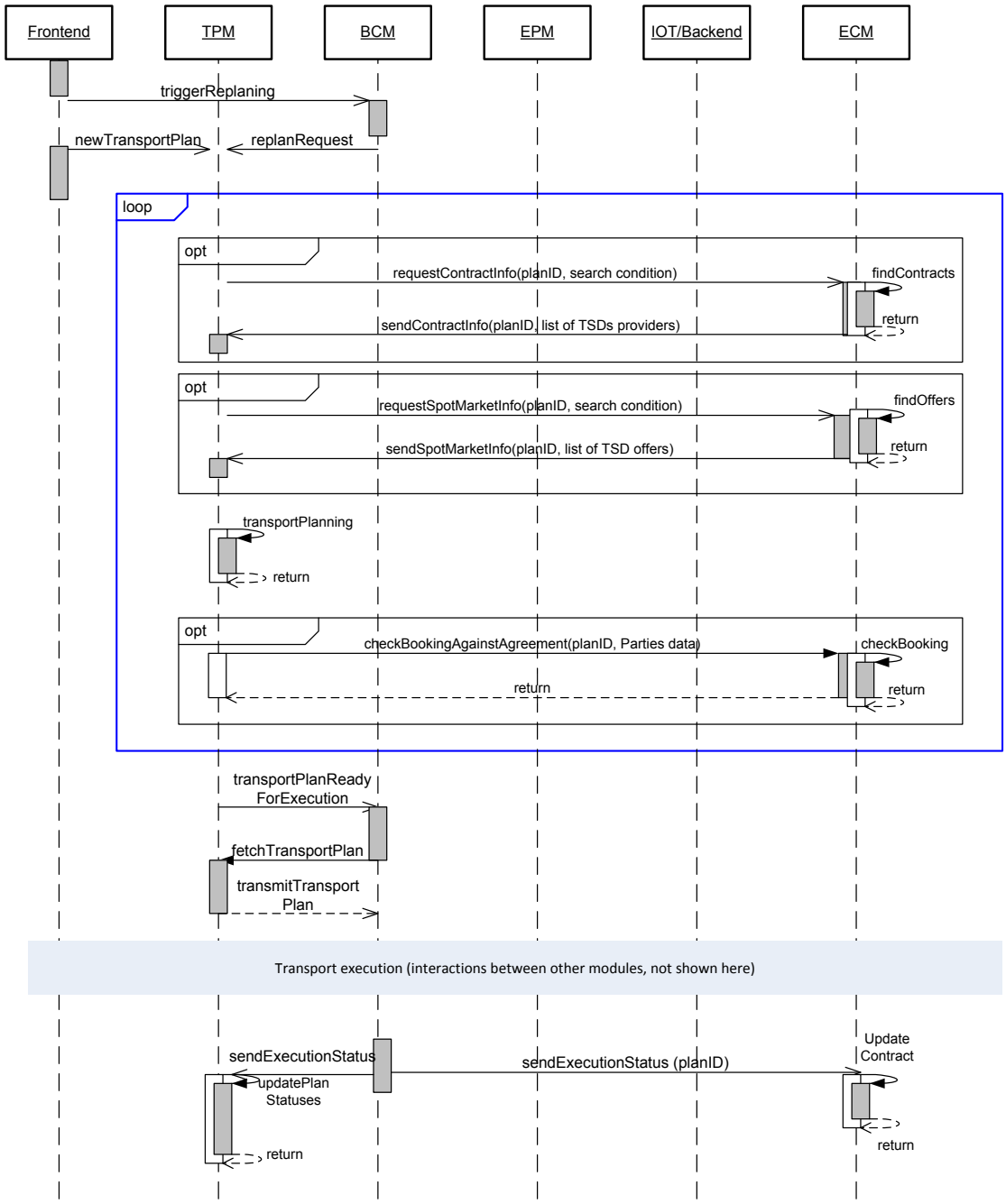

Figure 6 – Sequence diagram for interactions between the TPM and other modules

3.5. TPM Information Model

The information model of the Transport Planning Module is based on the information models of the Common Framework models [17], with the use of classes and structures from the Transport Service Description (TSD), Transport Execution Plan (TEP) and Goods Item Itinerary (GII) in the module. This section describes the overall data model. The section also describes the

Transport Chain Plan that is derived from the GII structure. Note that the TEP and TSD structures used by the TPM are close to the full structures described in e-Freight [16] and UBL2.1 [17], and will therefore not be described in this report.

3.5.1. Overall Data Model

Figure 7 shows the initial, overall information model used by the TPM for creation and maintenance of Transport Execution Plans. The classes are further described in Table 10.

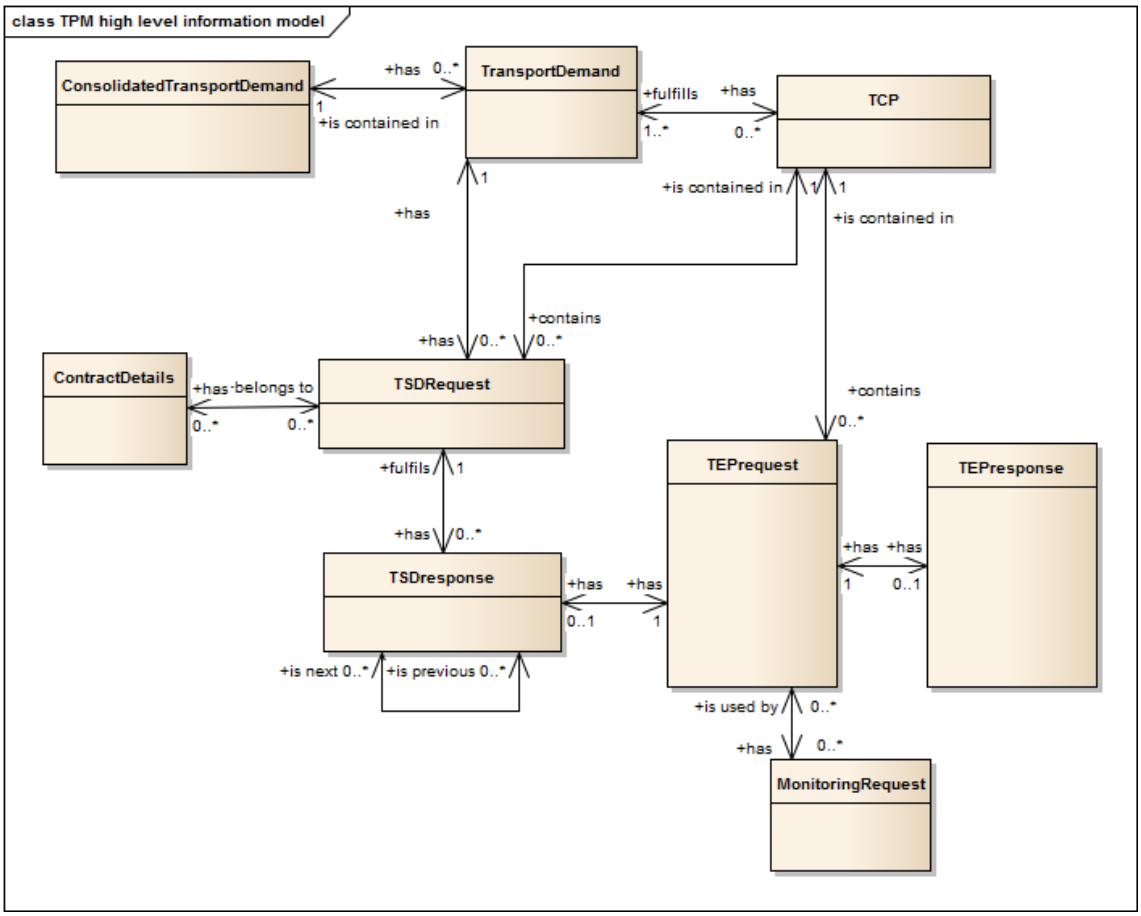

Figure 7 – Initial TPM Overall Data Model

Table 10 - Initial TPM Information Model Description

Class	Description
ConsolidateTransport-Demand	ConsolidatedTransportDemand is used to hold a set of transport demands that may have different (but still close) pickup locations or delivery locations, or different (but similar) pickup intervals or delivery intervals.

Class	Description
TransportDemand	TransportDemand contains a reference to all information dealing with this demand, both TCPs, TSDs, TEPs, and also monitoring information. In cases where a transport demand is updated, a new version is created. It also contains the TSDrequests that correspond to the original transport demand definition. When a new demand is added, a new instance of class TransportDemand is created. When an existing demand is updated, a new version of the TransportDemand is added and a new service search is done. Also, the status of this TransportDemand is set to 'Under Planning'. This means that a new or an updated transport demand will always lead to a new set of TSDrequests, TSDresponses, TCPs etc. The transport demand contains an id, description, and the creation date/time together with references to related transport data, and it also contains the initial TSDrequest describing this transport demand.
TCP	Class TCP – Transport Chain Plan – contains chains of TSDresponses and/or TEPrequests that may fulfill the related transport demand. Some or all of the TSDresponses will have a TEPrequest and/or TEPresponse. TCP can also contain TEPrequests and TEPresponses that do not have a TSD associated with it. Each TSDresponse is marked with a status saying whether it is part of an optimized set or not. Further, a status saying whether each TSDresponse is selected by the LSC or not, is added. If a transport demand is updated, a new service search is performed, and a new TCP instance is created.
TSDrequest	Class TSDrequest contains a set of TSDrequests that describes the TransportDemand where contract details from ECM has been added to the demand. For instance, this class may contain TSDrequests for specific LSPs that this LSC has a contract with. It also contains the original TSDrequest with no contract information added. The TSDrequest also contains requests sent to the Marketplace and to a LSP, for instance to a Resource Hub containing logistics services owned by several LSPs in a certain area.
TSDresponse	Class TSDresponse contains the set of TSDresponses received from the ECM or LSPs after a service search is done based on the associated TSDrequests.
TEPrequest	The class TEPrequest contains all versions of the TEPrequest sent from LSC to LSP. Each TEPrequest is related to zero or one version of a TSDresponse.
TEPresponse	The class TEPresponse contains all versions of the TEPresponses sent from LSP to LSC.
MonitoringRequest	Monitoring rules that are attached to a TEP.
ContractDetails	More contract details from the ECM can be stored in class ContractDetails, if needed.

3.5.2. Transport Chain Plan

The Transport Chain Plan (TCP) is an instantiation of the Goods Item Itinerary document from the Common Framework [17]. It is used to describe a transport chain, by referencing to the set of TEPs that transport chain constitutes of. In the internal information structure of the TPM, the TCP also refer to the TSD-requests representing parts of the chain that are yet unplanned.

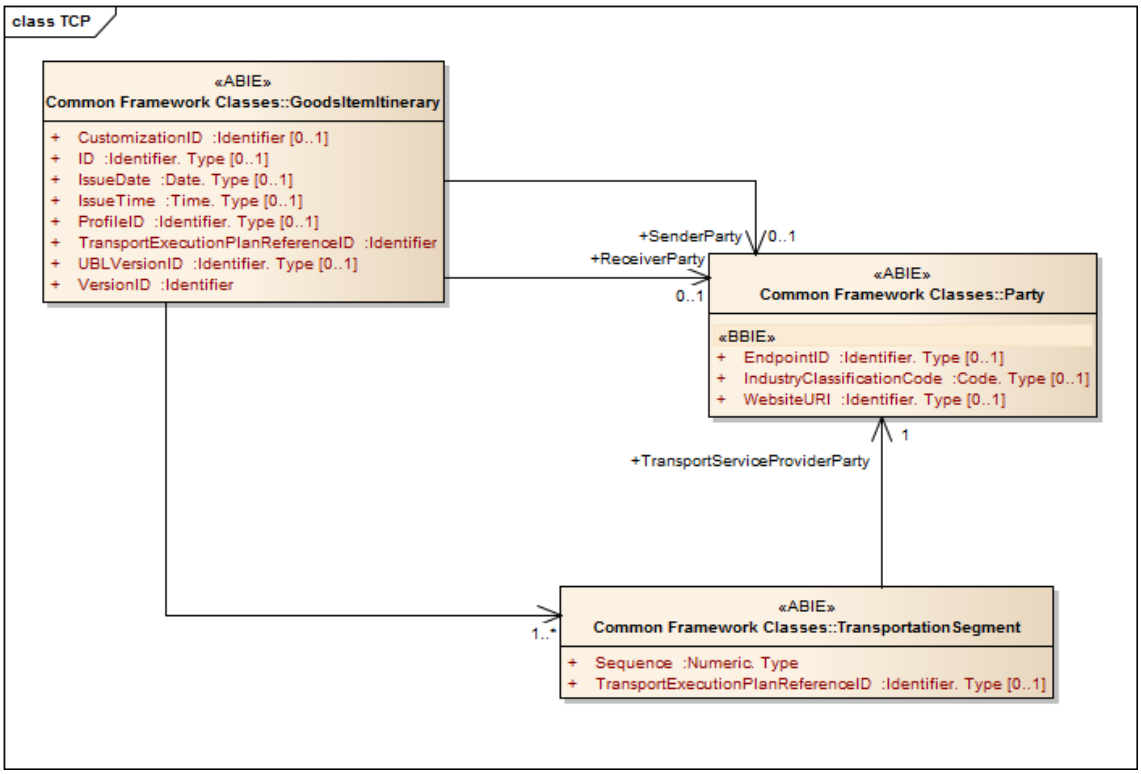

Figure 8 – TCP instantiated from GII

3.6. Use of Generic Enablers

It is expected that for creating some of the plans, large amounts of data must be processed and transferred, for instance when searching for different transport services that may fulfil the demand. The TPM may use Generic Enablers for module storage and the analysis of stored information within the storage; of the Generic Enablers investigated, the Data.BigData seems the most promising.

The Big Data GE uses technologies as data stream analysis, interrogating files and NoSQL storage to analyse and handle big data sets. Data stream analysis could potentially be used to analyse the Transport Service Descriptions received from the ECM or from a set of LSPs.

For the TPM Storage, the NoSQL ("Not Only SQL") storage seems promising. This is a document orientated storage system based on MongoDB, meaning it is handling document-oriented or semi-structured data instead of the tables and relations of relational databases. The

use of semi-structured data storage also seems useful with respect to the Common Framework information structures, allowing utilization of the in-built flexibility of the structures. The Big Data GE enables handling of data sets up to 0.5 Terabytes.

In the current design, no other Generic Enablers are used directly, but it is possible that the work with the detailed design and implementation of prototypes may reveal challenges that can be solved with the use of other Generic Enablers. It is also assumed that much of the required functionality that is provided through the FInest platform, like security and connections to legacy systems, will use Generic Enablers.

4. Summary and future work

This deliverable describes refinements to the requirements and the initial technical specification of the Transport Planning Module. The technical specification consists of models for component structure, interfaces and messages, and information structure of the TPM.

The component model assigns responsibilities for different parts of the described functionality to individual components or component groups, and describes the intended functionality of each component. The interfaces with other FInest modules, front-end applications and legacy systems have been further detailed, and the data types used in the information exchanges have been specified. The initial overall information model has also been specified in this deliverable.

In the final period of the project, the technical specifications will be further refined and detailed, with the aim of getting closer to specifications that can be used for a realization of the FInest modules.

The refinement work will go in parallel with the implementation of prototype demonstrators showing some of the intended functionality; it is assumed that the prototype implementation work results also will be used in the design work in order to identify strengths and weaknesses with the design, test the actual use of FIWARE Generic Enablers and get a better understanding of the interfaces and messages between the FInest modules.

References

[1] K. E. Fjørtoft, M. Hagaseth, L. S. Ramstad, Å. Tjora, C. Alias, M. Stollberg, M. Turkay and Y. Engel, “FInest D7.1 - Requirements analysis and selection of technology baseline for transport planning component,” 2011.

[2] M. Hagaseth, Å. Tjora, C. Alias, H. Koç, K. E. Fjørtoft, L. Ramstad, B. Özgür, C. Steinebach, M. P. Nowak, P. Koyuncu, S. Manisali, H. O. Yildirim, O. A. Kaptan and E.-J. van Harten, “FInest D7.2 - Conceptual Design of the Transport Planning Component,” 2012.

[3] J. T. Pedersen, P. Paganelli, F. Knoors and N. Meyer-Larsen, “One Common Framework for Information and Communication Systems in Transport and Logistics,” 2010.

[4] “eFreight Project Web Site,” [Online]. Available: http://www.efreightproject.eu/.

[5] “FreightWise project website,” [Online]. Available: http://freightwise.info/.

[6] “i-Cargo - Intelligent Cargo in Efficient and Sustainable Global Logistics Operations,” [Online]. Available: http://i-cargo.eu/.

[7] “DiSCwise project homepage,” [Online]. Available: http://www.discwise.eu/.

[8] M. Stollberg, R. Fleischhauer, M. Hagaseth, A. Metzger, C. Marquezan, G. Sharon, F. Fournier, Ö. Sönmezer and G. İşleyen, “FInest D3.3 - Interim Technical Specification for the Domain-Specific Future Internet (FI) Platform for Transport and Logistics,” 2012.

[9] R. Fleischhauer and S. Tschapke, “FInest D5.3 - Initial Technical Specification of Collaboration Manager Component,” 2012.

[10] F. Fournier, E. Rabinovich and G. Sharon, “FInest D6.3 - Initial Technical Specification of Event Processing Component,” 2012.

[11] C. Marquezan, S. Heyne, M. Stollberg, N. Golmohamadi, A. Metzger, M. Hagaseth, Å. Tjora, C. Alias, Ø. Olsen, M. Zahlman and M. Turkay, “FInest D8.3 - Initial Technical Specification of Logistics Contract Manager,” 2012.

[12] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns - Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[13] R. Fleischhauer and S. Tschapke, “FInest D5.2 - Conceptual Design of Collaboration Manager Component,” 2012.

[14] Y. Engel, G. Sharon, F. Fournier, Å. Tjora, M. Zahlmann, T. T. Knutsen and E.-J. van Harten, “FInest D6.2 - Conceptual Design of Event Processing Component,” 2012.

[15] C. Marquezan, S. Heyne, C. Alias, Ø. Olsen, M. Turkay, A. Koestler, M. Zahlmann and N. Golmohamadi, “FInest D8.2 - Conceptual Design of Logistics Contract Manager,” 2012.

[16] A. Vennesland, H. Westerheim, M. Rosén, C. Kjellberg, V. Mary, P. Wiman and J. T. Pedersen, “e-Freight D1.3b: e-Freight Framework - Information Models,” 2010.

[17] “UBL 2.1,” [Online]. Available: http://www.oasis-

open.org/committees/tc_home.php?wg_abbrev=ubl. [Accessed 19 March 2012].