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

D7.5

Final Technical Specification and Phase 2 Implementation Plan for 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	
Editor	Marianne Hagaseth	MARINTEK
Contributors	Åsmund Tjora	MARINTEK
	Marianne Hagaseth	MARINTEK
	Christian Steinebach	MARINTEK
	Rene Fleischhauer	SAP
	Clarissa Marquezan	UDE
	Fabiana Fournier	IBM
Reviewers	Bülent Erbaş	KOÇ
	Rene Fleischhauer	SAP
Dissemination Level	PU	
Contractual Delivery Date	30-03-2013	
Actual Delivery Date	30-03-2013	
Version	1.0	

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 fifth deliverable from FInest work package 7, "Transport Planning and Replanning"*, and contains the final version of the specification of the Transport Planning Module developed in the FInest project, in addition to the plans for further development of the Module concept as the "Logistics Planning app" in the Phase 2 cSpace project.*

The final specification of the module is based on the initial specification described in the third deliverable from this work package, "Initial Technical Specification of the Transport Planning Component"*, with corrections and clarifications based on lessons learned with the work on the prototype implementation of the Module.*

The specification consists of models describing the component structure of the Transport Planning Module, the description of the interfaces both for use by other FInest modules and external users and systems, an information model describing the system's internal database structure, as well as data types used in the interfaces, including the Module's use of the UBL common framework structures for transport execution plans and transport service descriptions.

With this document, we complete task T7.2 "Conceptual Design and Technical Specification of the Transport Planning Component" and task T7.5 "Phase 2 Implementation Plan of the Transport Planning Component". Task T7.1 "Requirements Analysis and Selection of Technology Baseline for the Transport Planning Component" and task T7.3 "Technological Alignment with FI PPP Core Platform" are reported in Deliverables D7.1, D7.2 and D7.3.

Task T7.4 "Prototypical Implementation of the Transport Planning Component" is reported in Deliverable D7.4.

Document History

Version	Date	Comments
V0.1	18.10.2012	Initial version
V0.2	14.02.2013	Preliminary version for plenary meeting
V0.3	06.03.2013	Version for internal review
V1.0	30.03.2013	Final version

Table of Contents

List of Figures

List of Tables

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 final technical specification for the FInest Transport Planning Module (TPM), including the component structure, interfaces, message types, and internal information model.

The work is mainly a refinement of the specifications reported in FInest deliverable D7.3, based on the work on the module design and feedback from the prototype implementation work (the prototype is delivered as deliverable D7.4).

The deliverable also contains a preliminary plan for the implementation and further development of the module for Phase II of the FI-PPP program.

1.1. Brief description of the Transport Planning Module

The Transport Planning Module is a Future Internet module offering services for planning transport chains and services, including finding transportation services matching the demand, making use of existing contract details, and setting up bookings. Its functionality is roughly divided into 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 E-Contracting Module (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 of the services between the logistics service provider (LSP) and the logistics service client (LSC) and the booking of the services. When a plan changes status from "under development" to "ready for execution", it will be transferred to the Business Collaboration Module (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-requests) and Transport Service Descriptions (TSD) on external systems. The messages used will conform to the Common Framework model [1]. The Common Framework messages are also used in other EU transport projects, e.g. e-Freight [2], FreightWise [3], i-Cargo [4], DiSCwise [5] and others.

1.2. Other Work on the Transport Planning Module

Earlier work on the TPM has been documented in the deliverables D7.1, D7.2 and D7.3. A prototype, deliverable D7.4, has been implemented concurrently with the work on the final technical specification.

Deliverable *D7.1 – Requirements Analysis and Selection of Technology Baseline for Transport Planning Component* [6] 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* [7] 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.

Deliverable *D7.3 – Initial Technical Specification of the Transport Planning Component* [8] uses the scope and requirements of D7.1 and the conceptual design from D7.2 as a base for describing a component architecture for the TPM, giving an initial definition of the interfaces with other FInest modules, front-end and other systems, and describing the initial data model for the module.

Deliverable *D7.4 – Prototypical Implementation of the Transport Planning Component* [9] is a prototype implementation of some of the main features of the Transport Planning Module.

The technical specifications in D7.3 serve as the main base for the work in this deliverable, but the scope, requirements and functionality descriptions in D7.1 and D7.2 is of course still used as reference for the work. The work on the specifications has been done in cooperation with the implementation of the prototype delivered as D7.4. The specification work has functioned as the main base for the prototype implementation, and the implementation work also has given important feedback for the specifications in this deliverable.

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 7 on TPM development is tightly connected to the other technical work packages 5, 6 and 8, Business Network Collaboration, Proactive Event Driven Monitoring and Logistics Contract Establishment and Management. The Prototypical implementation of each of these modules has been used to set up a story line to demonstrate the intended FInest capabilities.

The work done with the TPM also fits in to what has been conducted in work package 3, Solution Design and Technical Architecture, providing the overall design and architecture for the FInest platform.

In addition to this, the use case specification done in work package 2 has served as a refinement of requirements, presentation of domain challenges, as well as experiment specifications for domain trials, and this has been an important input to the TPM development.

1.4. Structure of this deliverable

The rest of the deliverable is structured as follows:

Chapter 2 describes the technical architecture of the Transport Planning Module at the end of the FInest project, including component structure and internal information model.

Chapter 3 describes which generic enablers are relevant for the TPM.

Chapter 4 describes the implementation plans for the cSpace project's continuation of the work described in FInest's WP7, i.e. the cSpace "Logistics Planning app".

Chapter 5 summarizes and concludes the report.

Appendices 6 and 7 give more details of the Transport Execution Plan and Transport Service Description structures used by the Transport Planning Module.

1.5. Differences to Deliverable D7.3

In order make this deliverable a self-contained document, the unchanged parts from D7.3 are repeated here. Based on this, changes made to form the final technical specification are described in this deliverable. The changes between D7.3 and D7.5 are basically as follows:

- Section 2.1:
  - Added component "TSD Rule checker" in component diagram in Figure 1.
  - Section 2.1.2.2: New
  - Rest of Section 2.1: No changes in functionality, only some additions to the descriptions of the components.
- Section 2.2: No changes
- Section 2.3:
  - Section 2.3.1: New
  - Section 2.3.2: Unchanged
  - Section 2.3.3: Reference to TCP added to ReplanningType in Figure 6.
- Section 2.3.4: New
- 2.3.5: New
- Section 2.3.6 : Simplifications done in the model, Figure 13 TCP data model
- Section 2.4: New

- Section 2.5: Sequence diagram updated with more details on booking and replanning
- Section 2.6: New
- Chapter 3: No changes
- Chapter 4:
    - New

2. Final Technical Architecture of Transport Planning Module

This chapter contains the technical architecture of the Transport Planning Module for the FInest project. While the results presented here are "final" in the sense of the FInest project, and are believed to fulfil the main concept, requirements and system description of earlier WP7 deliverables [6] [7], the design should also be open for changes and improvements, e.g. in the scope of the follow-up cSpace project.

Large parts of the architecture are unchanged from the D7.3 deliverable [8]; but for the sake of overview, the unchanged parts are also included in this deliverable.

2.1. Transport Planning Module Component Model

The overall structure of the design described in D7.3 [8] is kept in the version presented in this deliverable; there has not been detected any reason to change this structure.

It may, however, be useful to further detail the design of some of the components described in order to break down components with complex behaviour to a set of simpler components. As the implemented parts of the prototype only make up a small subset of the full system, such detailing has not been done at this point, but should be considered when implementing a full system. Figure 1 shows the component structure.

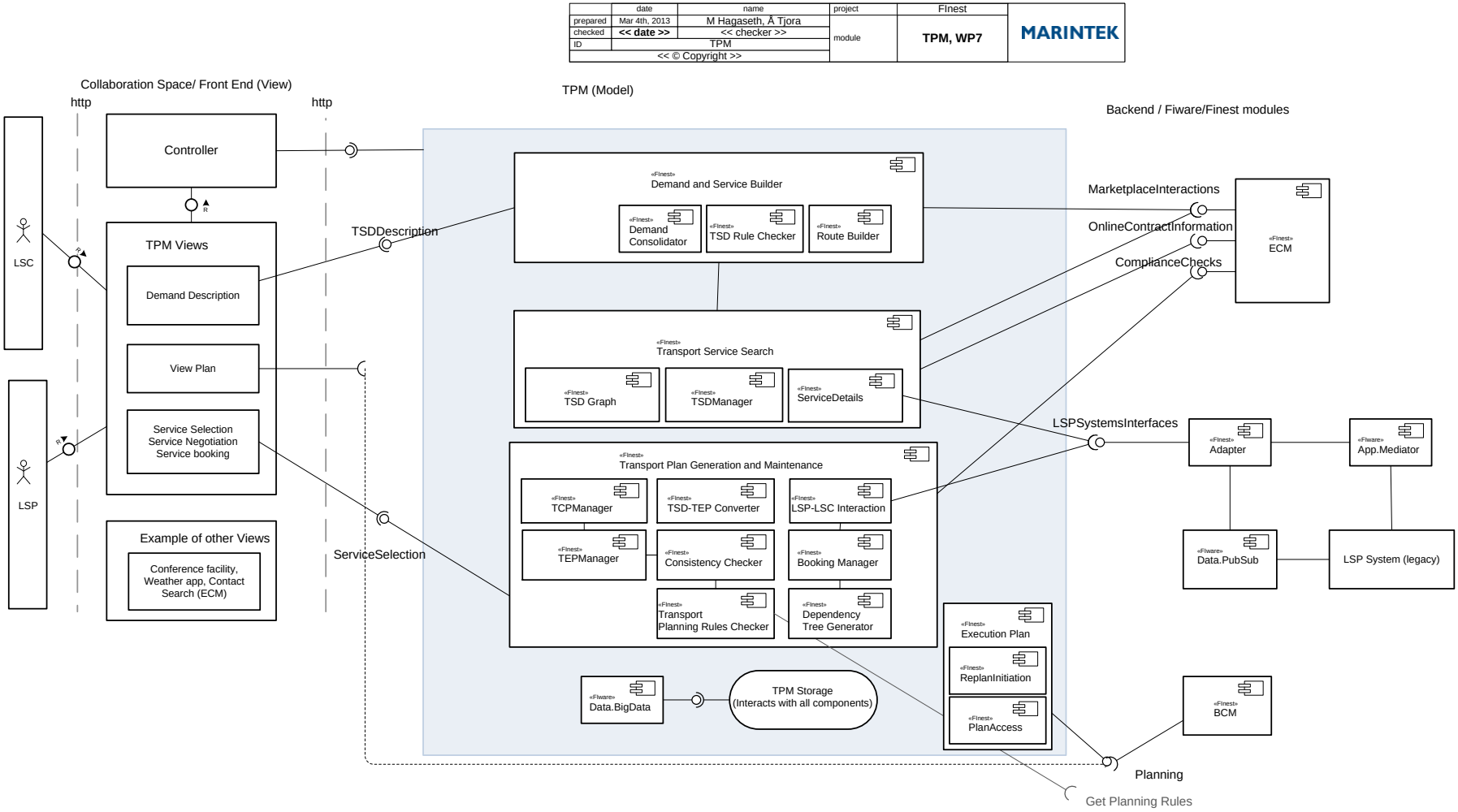
date
name
project
FInest
prepared
Mar 4th, 2013
M Hagaseth, Å Tjora
checked
<< date >>
<< checker >>
module
TPM, WP7
MARINTEK
ID
TPM
<< © Copyright >>
Collaboration Space/ Front End (View)
http
http
TPM (Model)
Backend / Fiware/Finest modules
Controller
LSC
LSP
TPM Views
Demand Description
View Plan
Service Selection
Service Negotiation
Service booking
Example of other Views
Conference facility, Weather app, Contact Search (ECM)
TSDDescription
ServiceSelection
«Finest» Demand and Service Builder
«Finest» Demand Consolidator
«Finest» TSD Rule Checker
«Finest» Route Builder
«Finest» Transport Service Search
«Finest» TSD Graph
«Finest» TSDManager
«Finest» ServiceDetails
«Finest» Transport Plan Generation and Maintenance
«Finest» TCPManager
«Finest» TSD-TEP Converter
«Finest» LSP-LSC Interaction
«Finest» TEPManager
«Finest» Consistency Checker
«Finest» Booking Manager
«Finest» Transport Planning Rules Checker
«Finest» Dependency Tree Generator
«Fiware» Data.BigData
TPM Storage (Interacts with all components)
«Finest» Execution Plan
«Finest» ReplanInitiation
«Finest» PlanAccess
MarketplaceInteractions
OnlineContractInformation
ComplianceChecks
«Finest» ECM
LSPSystemsInterfaces
«Finest» Adapter
«Fiware» App.Mediator
«Fiware» Data.PubSub
LSP System (legacy)
«Finest» BCM
Planning
Get Planning Rules

Figure 1 Component Model for the Transport Planning Module

2.1.1. Changes in Component Model

As a result of reviewing and working with the component structure during the prototype implementation, some name changes to the components described in D7.3 [8] was suggested. These changes are mainly to get a more consistent naming convention; they do not change the components' behaviour. Also, a new component is added (TSD Graph).

2.1.2. 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.

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 consolidator")

2.1.2.1. Route builder

This component offers functionality for describing the route that the goods will take for fulfilling a demand. The interfaces for this subcomponent should be accessible both for manual route building and automatic route building tools, meaning that the waypoints can be either entered manually by the user or fetched automatically..

The component allows to enter a set of possible waypoints between the start and end location of the transport. Based on these locations, a set of TSD requests representing service requests between all possible locations are created. This means that the output of the route builder is a set of TSD requests each describing one requested service in the given transport demand.

After a transport demand has been transformed to a set of TSDrequests, each of the TSD requests can be sent to the ECM to search for actual services and contracts, as described in Section 2.1.3.2.

Figure 2 shows as an example of route building for a transport demand from Fiskarstrand, Norway to Rotterdam, Netherlands. When using this component, both the start location

(Fiskarstrand), the end location (Rotterdam), and a waypoint (Ålesund) is given. The edges are TSD requests sent to the ECM to search for relevant contracts and services for each of the leg.

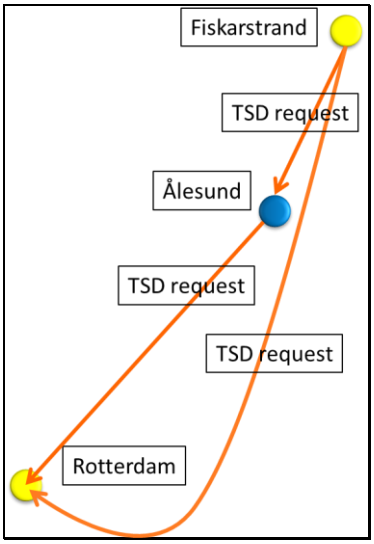

Figure 2 Route Building Example

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.

2.1.2.2. TSD Rule checker

This component handles rule checking for building valid service and demand descriptions, and detects missing information. The criterion is the minimum information to be able to start planning with the TSD. Basically, this is rules to check the validity of the given transport demand.

Summary of functionality:

- Check that a TSDrequest fulfils a minimum set of information.
- Check that a TSD response fulfils a minimum set of information. This is relevant for LSPs setting up services offers in the TPM.
- 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.

2.1.2.3. Demand consolidator

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.

2.1.3. 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, that is, in the LSP's systems.
- **Search for services on spot market:** The search for relevant TSD responses will be done at the spot market. After a TSD response has been found this way, the component will try to update the TSD response information by sending requests to the LSP's systems.
- **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, for instance when a LSP wants to add service descriptions directly to the TPM.

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 "Service details" component)
- Storage of a pool of TSDs that can be used for fulfilling the demand
- Generation of a graph of the stored TSD responses that will be used for finding the best overall transport solution.
- Maintenance on the pool of TSDs so that they are reasonably updated (delegated to "TSD Manager")

2.1.3.1. TSD Graph

This component offers functionality for setting up a directed graph consisting of TSDresponses fulfilling a given transport demand. The input to the TSD graph builder is a transport demand consisting of all cargo data, and the initial and final locations, and other information defining the transport demand. Then, this component searches the ECM for possible services and contracts using functionality in the "Service details" component described in Section 2.1.3.2 to find possible services to fulfil the transport demand.

This component also has functionality for specifying service searches from a start location, where a recursive search can be performed to find offered transport services through waypoints not specified by the "route builder" component. As the component uses the "Service Details" component functionality, the waypoints generated this way will be based on actual offered services. The directed graph of TSD responses created by this component can be used directly in the selection process and further in the booking process (Section 2.1.4.8).

To limit the searching time and the needed interaction with the ECM, restrictions can be made for instance on the maximum total price, the maximum time consumption, or the maximum number of legs/services to fulfil a transport demand.

The component also offers functionality to find the best route through the graph, based on some given priorities (e.g. time, service prices etc.) from the user.

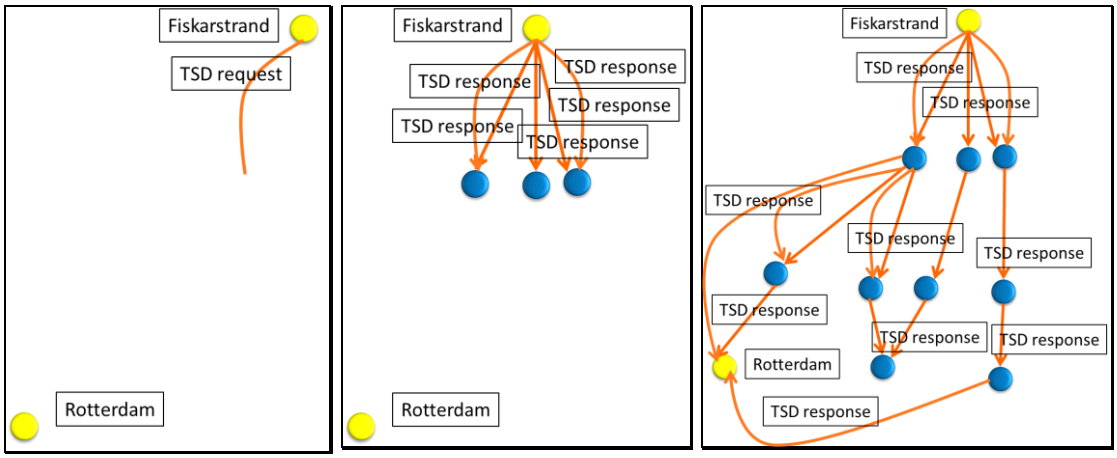

Figure 3 TSD Graph Example

For the graph shown in Figure 3, only the start location Fiskarstrand and the end location Rotterdam is given. Each edge in the final graph is a TSDresponse fetched from the ECM as a response to service and contract searches done, starting from Fiskarstrand and repeating the search from there.

Summary of functionality:

- Build a directed graph of TSD responses from the start to the end location.
- Find not only the best route, but a list of possible routes sorted by some cost function, for instance price or time. This means that some algorithm must be used to find the Kth shortest path in the directed graph from the start location to the end location, for instance the one used in the TPM prototype [10], [11].

2.1.3.2. 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 (LSP systems)

- Request subscription to updates of relevant TSDs
- Find relevant TSDs in user-specific storage

2.1.3.3. TSD Manager

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. These updates may be needed because of changes in the service descriptions on the LSP's side that have occurred after the previous service search was finished and before booking negotiation is started.
- Mark TSDs or remove them from pool when they are no longer usable to fulfil the request.

2.1.4. 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 Plan requests, 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 "TCP Manager")
- Convert TSDs describing services and demand as well as known contract information to an initial TEP request (delegated to "TSD-TEP Converter")
- 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 Manager").
- Apply location-, goods- and mode-specific rules to the consistency check and TEP configuration (delegated to "Transport Planning Rules Checker")

- 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 Generator")
- Support the negotiation and booking of TEPs (delegated to "LSP-LSC Interaction")

2.1.4.1. TSD-TEP Converter

This subcomponent is responsible for creating Transport Execution Plan requests based on the Transport Service Descriptions responses 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 request.
- Add data from TSD-request (demand) to TEP request
- Add data from TSD response (service) to TEP request
- Add contract information to TEP request
- Place TEP reference in the TCP (call to "TCP Manager")
- Place TCP reference in the TEP

2.1.4.2. TCP Manager

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
- Add reference to the TCP in the TEP
- 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.

2.1.4.3. 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. Examples of rules are to check that for sequential services, the second service starts after the previous one has completed, or check if there is consistency regarding the start and end locations.

Summary of functionality:

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

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

2.1.4.4. TEP Manager

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.

2.1.4.5. Transport Planning Rules Checker

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 rule sets used are external to the TPM, and that this component fetches relevant rule sets from external sources. An example of a rule for the fish transport from Fiskarstrand, Norway to Rotterdam, Netherlands, and further to Brazil is a rule added to a document handling service saying that the import licence to Brazil must be ready latest 24 hours before the fish is loaded on board the feedering vessel in Ålesund. Another example is that different rules must be applied for import/export to different countries/regions.

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

2.1.4.6. Dependency Tree Generator

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, Section 2.1.4.8. An example is the case where a fish transport from Fiskarstrand, Norway to Rotterdam, Netherlands is to be booked. Then, the feedering service from Ålesund to Rotterdam should be booked first, before the truck transportation from Fiskarstrand to Ålesund is booked.

Summary of functionality:

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

2.1.4.7. LSP-LSC Interaction

This subcomponent is responsible for the contact between the provider and client in a TEP negotiation and booking process for one TEP. The functionality should have support for using a two stage booking process, including a first stage where reservation of the service booking is done, and a second stage where the service booking is confirmed. Some details on this TEP negotiation is further described in Section 2.3.4.1.

For booking systems not offering the possibility to do reservations, the negotiation process will end with either a confirmed booking or a not-confirmed booking.

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).

2.1.4.8. Booking Manager

This subcomponent will manage the negotiation and booking process for a complete TCP consisting of one phase where each LSP is asked to do a reservation, and a second phase where each booking is confirmed. It must include the possibility to confirm the booking of all TEPs in a TCP when the negotiation of all TEPs between the LSC and the LSPs has been completed. In addition, if at least one of the TEP negotiations fails, the reservations made for all other TEPs must be cancelled, and the user must be notified. If the LSP's booking system does not have the possibility to do reservations, the booking must be confirmed and later be cancelled. However, the use of the "Dependency Tree Generator" (Section 2.1.4.6) should reduce the number of such cases, for instance by ensuring that the bookings are done in a certain sequence.

Summary of functionality:

- Keep track of the negotiation and booking progress for the TEPs in a TCP, including keeping logs of the interactions (negotiation processes including several iterations).
- Management of automated booking processes consisting of only one request from the LSC and one response from the LSP, including both standard bookings (only confirmed/non-confirmed bookings) and also two stage bookings (confirmed/reserved/non-confirmed bookings).

2.1.5. Supporting components

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

2.1.5.1. 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 2.3.

2.1.5.2. Execution plan

This component handles tasks and interfaces related to several of the main components, and it is mainly used for accessing the plans and the communication with the transport execution environment (i.e. the BCM). Also, the ECM needs access to the finalized transport plans to be able use updated booking information to maintain contracts and service descriptions.

The two main subcomponents are the "Replan Initiation" and "Plan Access" components.

2.1.5.3. Replan Initiation

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.

2.1.5.4. Plan Access

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

2.2. Transport Planning Module Interfaces

The TPM needs interfaces to the ECM for contract details, marketplace connections, compliancy checks, and to send transport plans, to the BCM for transferring the finished plans to execution and for receiving replanning triggers. The module also must 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.

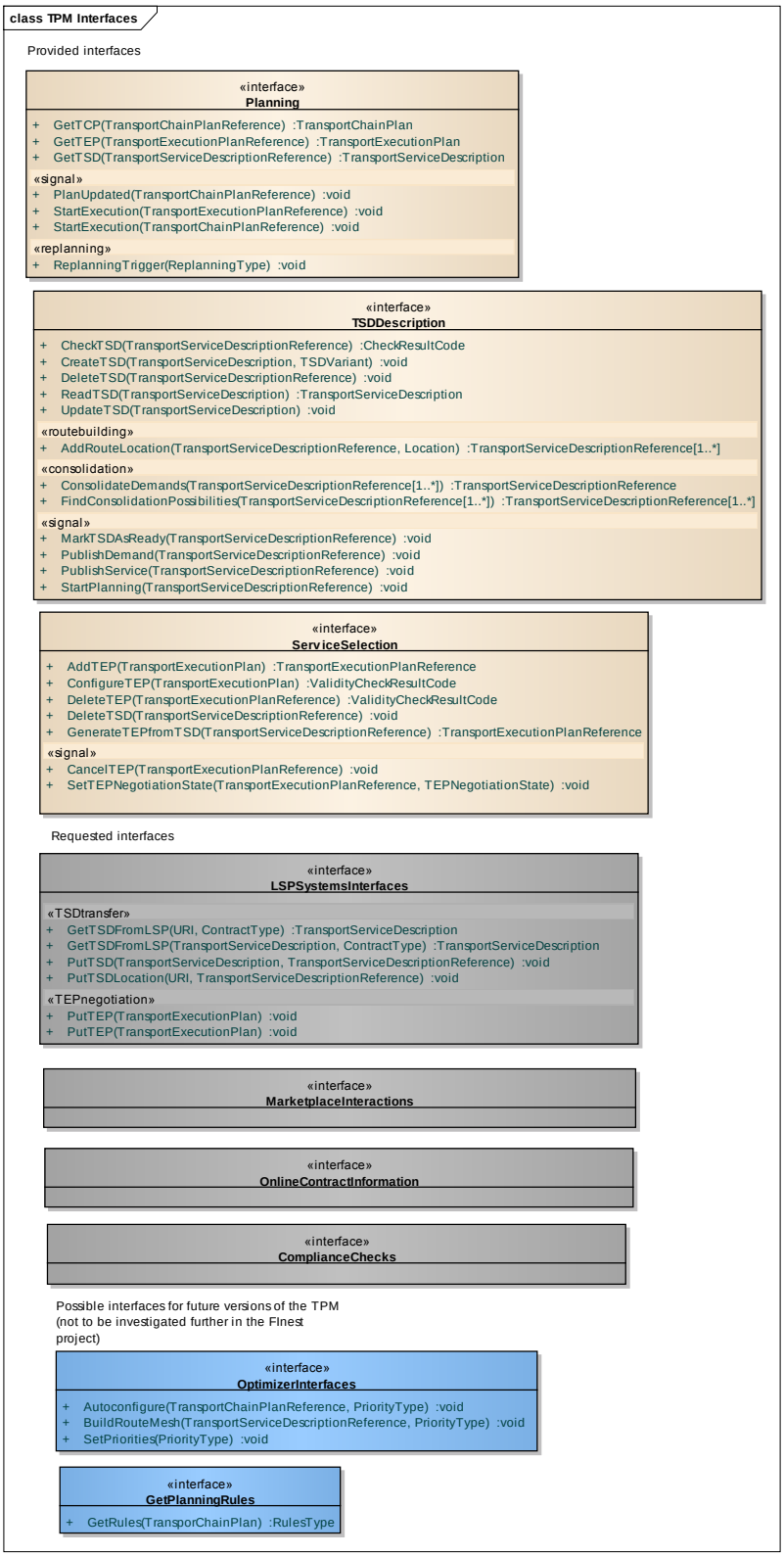

Figure 4 TPM Interfaces

2.2.1. 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 interface descriptions are found in deliverable D8.5 [12]

2.2.2. GetPlanningRules (optional)

This is a 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 current TPM design, meaning that it is optional for the TPM to use such an interface provided by some other. The design should however be open for this option in future versions.

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

Table 1 GetPlanningRules interface

2.2.3. LSPSystemsInterfaces

This is an interface for contact with logistic service 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.

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 Public	Sends a TEP to LSP for negotiation (i.e. TPM is the sender)	**TransportExecutionPlan** [in] TEP

Method	Notes	Parameters
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. This is used to be able to fetch a service description based on a URI of the TSD.	**URI** [in] TSDLocation **TransportServiceDescriptionReference** [in] DemandReference

Table 2 LSPSystemsInterfaces interface

2.2.4. OptimizerInterfaces (optional)

The TPM can have a possible interface to optimizing components.

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

This is an optional part of the current TPM design. The design should however be open for this option in future versions.

Method	Notes	Parameters
Autoconfigure() void Public	Initiates auto-configuration 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

Table 3 OptimizerInterfaces interface

2.2.5. Planning

This is an interface to the plan execution facilities and plan viewers.

The view plan methods will only return the parts known by the 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 the TPM and the BCM.

Method	Notes	Parameters

Method	Notes	Parameters
GetTCP() TransportChainPlan Public	Returns the Transport Chain Plan (containing TEPs and possibly 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 the BCM and the ECM) 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 the demand and searches 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. These are signals sent from the front end to the TPM.	**TransportExecutionPlanReference** [in] TEPReference
StartExecution() void Public	Signals to the TPM that a chain plan is ready for execution. These are signals sent from the front end to the TPM.	**TransportChainPlanReference** [in] TCPReference

Table 4 Planning interface

2.2.6. ServiceSelection

This is an 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), and "Contract Issues" (meaning that the updates of the TEP may violate a contract between a LSC and a LSP).

Method	Notes	Parameters
AddTEP() TransportExecutionPlanReference	User adds a TEP (i.e. not generated by system)	**TransportExecutionPlan** [in] TEP

Method	Notes	Parameters
Public		
CancelTEP() void Public	Cancels a TEP that has entered the negotiation process or has been booked. The TPM must contact the negotiation partner.	**TransportExecutionPlanReference** [in] TEPReference
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, that is, from a LSP or a marketplace).	**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 (cancelled, planned, rejected, confirmed, not confirmed, completed, disputed, revised)	**TransportExecutionPlanReference** [in] TEPReference **TEPNegotiationState** [in] State

Table 5 ServiceSelection interface

2.2.7. TSDDescription

This is an interface for performing operations on the Transport Service Descriptions.

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

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()	Checks the completeness and validity of a	**TransportServiceDescriptionRefe**

Method	Notes	Parameters
CheckResultCode Public	TSD, e.g. whether the TSD can be used for planning as it is, or if more information must be submitted.	**rence** [in] TSDreference
ConsolidateDemands() TransportServiceDescriptionReference Public	Consolidates a set of demands.	**TransportServiceDescriptionReference[1..*]** [in] Demands
CreateTSD() void Public	Create a TSD for either a demand or a service request	**TransportServiceDescription** [in] TSD **TSDVariant** [in] ServiceOrDemand
DeleteTSD() void Public	Delete a TSD	**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	Read a TSD from the TPM storage	**TransportServiceDescription** [in] TSDReference
StartPlanning() void Public	Starts the planning process with the given demand	**TransportServiceDescriptionReference** [in] TSDReference
UpdateTSD() void Public	Update values of a given TSD	**TransportServiceDescription** [in] TSD

Table 6 TSDDescription interface

2.3. Transport Planning Module Information Model

2.3.1. Relation to Common Framework and UBL Standards

The FInest information model regarding TEP, TSD and TCP is based on the UBL standard model version 2.1 [13]. The starting point for the FInest model has been the core profile defined in the eFreight [14]. Some extensions have been done to that model, and additions have been made to the UBL standard model, see Section 2.4.

2.3.2. Internal TPM Information Model

There are no changes to the overall information model compared to what was explained in D7.3 [8]. The model is repeated in Figure 5 for completeness. However, a lot of refinements on the details of each of the classes are done, as further described in the rest of this section.

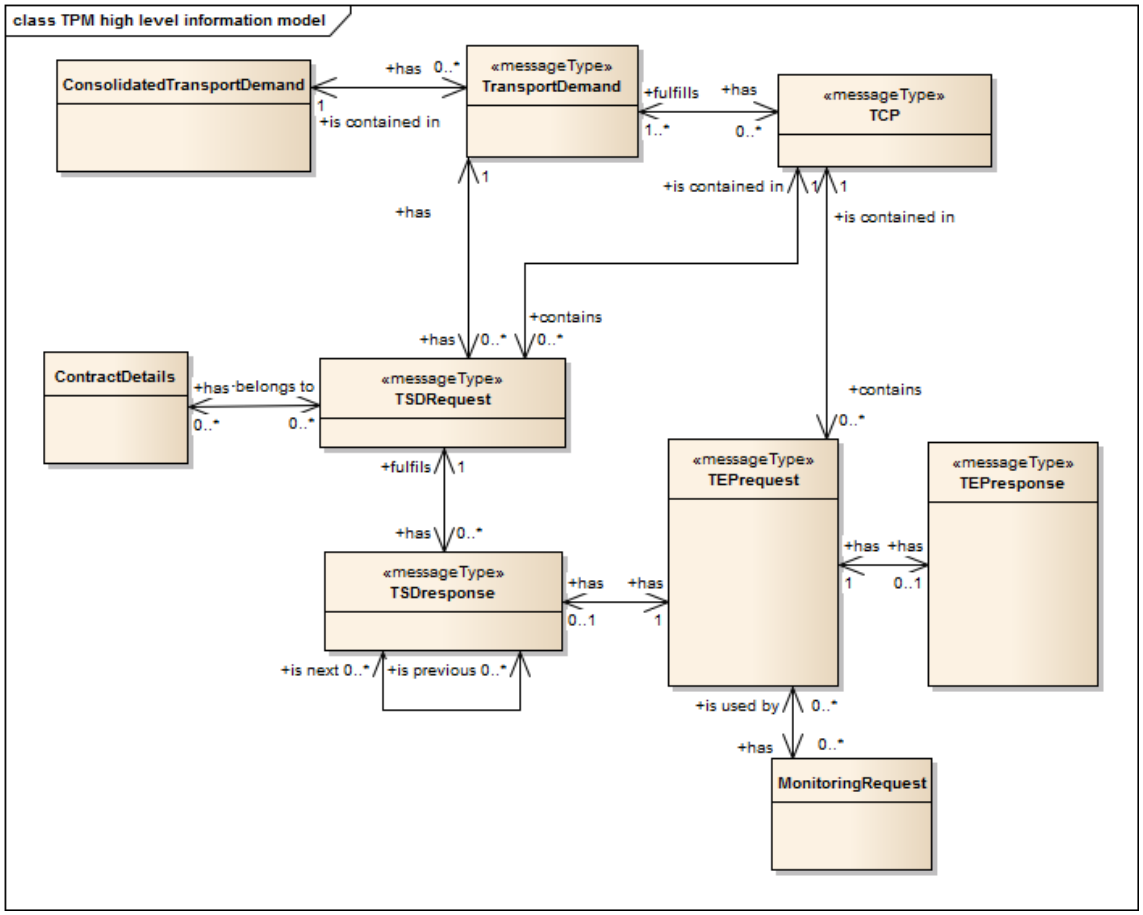

Figure 5 TPM Overall Data Model

2.3.3. Information Model for TPM Interfaces

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 [14], the messages are also part of UBL 2.1 [13]. The Transport Chain Plan (TCP) data type is based on the Goods Item Itinerary (GII) message also described in UBL 2.1In addition, the TPM will make use of the data types defined by deliverable D8.5 [12] for the interfaces with the ECM.

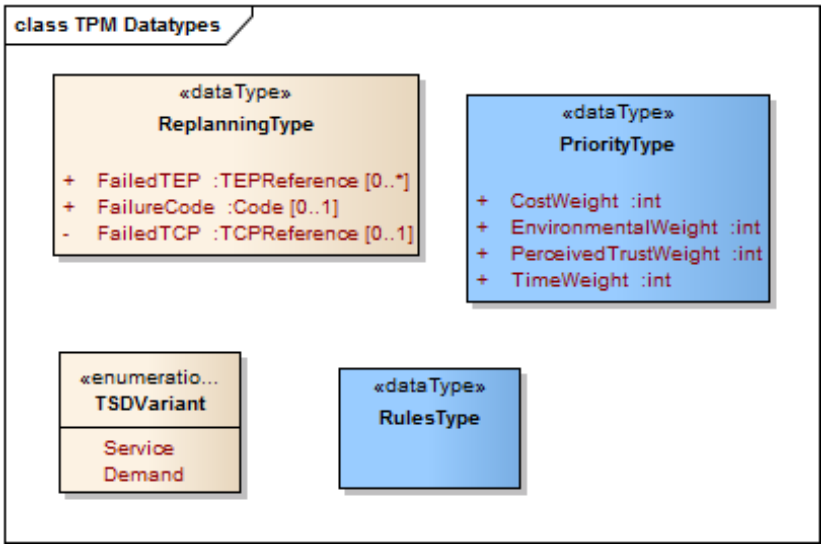

Figure 6 TPM Interface Data types

2.3.3.1. TSDVariant

TSDVariant gives an 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.

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:*

Table 7 TSDVariant

2.3.3.2. 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 auto configuration is outside the scope of the TPM during the FInest project, this data type is just an example, and may be changed.

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:*

Table 8 PriorityType

2.3.3.3. ReplanningType

This data type is used for giving 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.

Attribute	Notes	Constraints and tags
FailedTEP TEPReference [0..*] Public	Refer to one or more TEPs that have (or potentially will) fail.	*Default:*
FailureCode Code [0..1] Public	Gives a status code for the TEP failures. The coding 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:*

Attribute	Notes	Constraints and tags
FailedTCP TCPReference Public [0..1]	A reference to the TCP which needs to be replanned.	*Default:*

Table 9 ReplanningType

2.3.3.4. RulesType

This is a 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 will be specified in later versions.

2.3.4. Details on TEP

Figure 7 shows the TEP response information model as it is used in the TPM design, and Figure 8 shows the TEP request information model. The grey classes are further described in Appendix 6.

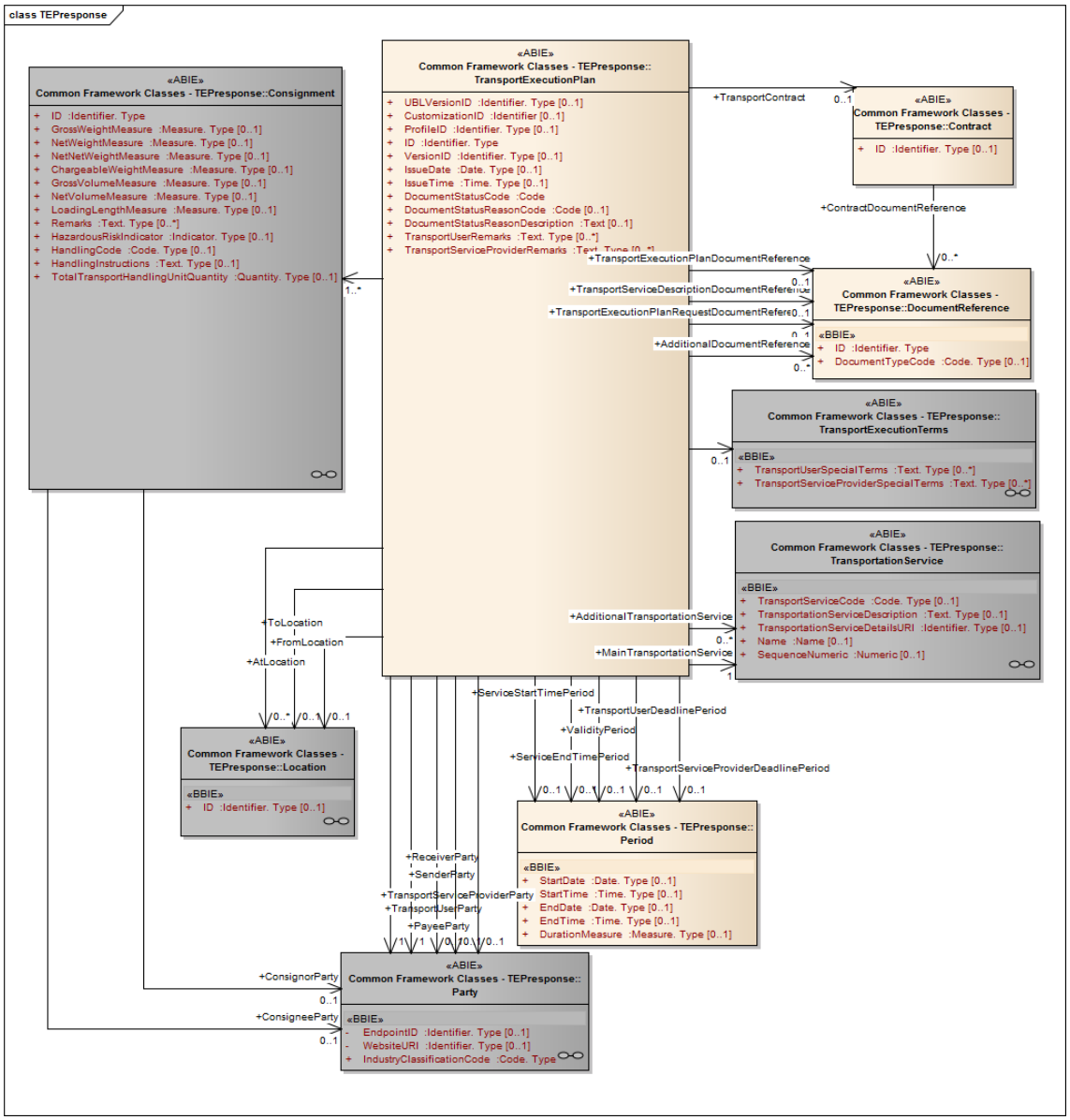

Figure 7 TEP Response

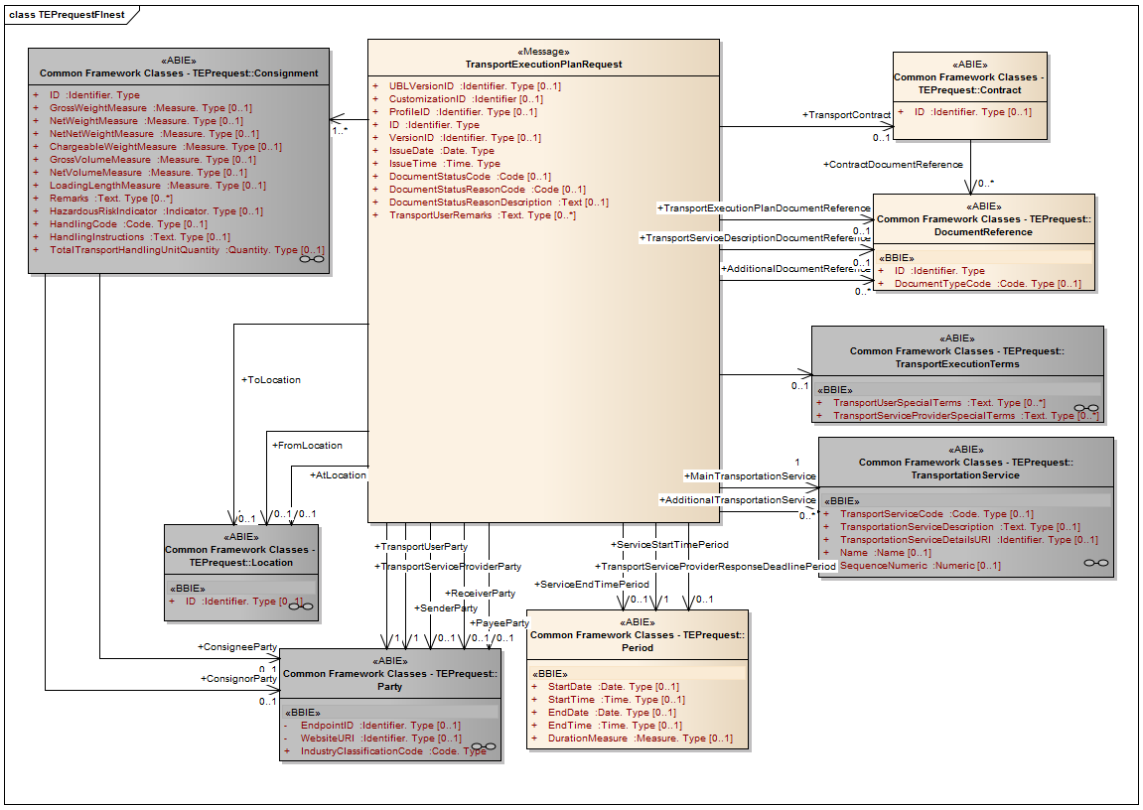

Figure 8 TEP Request

2.3.4.1. Details on TEP Identification

The following describes the principles of the TEP negotiation [14]. The identification of the TEP is done as follows, where the TPM in this case has the role of a LSC, Figure 9.

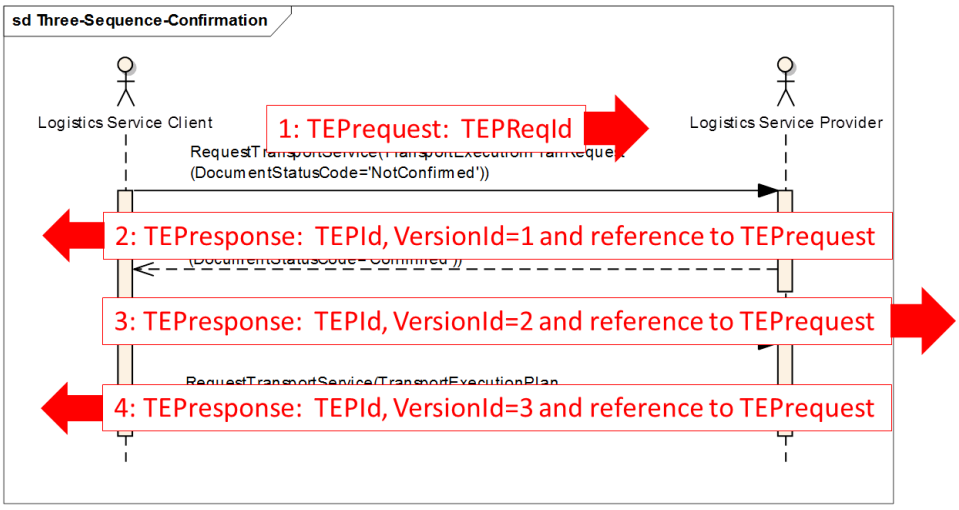

Figure 9 TEP Identification

1) LSC sends TEP Request with *TEPid*="TEP-req-1-1"

2) LSP sends a response TEP with *TEPid*="TEP1-1" and VersionId=1, and also with a reference to "TEP-req-1-1".
3) LSC sends a response TEP with *TEPid*="TEP1-1" and *VersionId*=2, and also with a reference to "TEP-req-1-1".
4) LSP sends a response TEP with *TEPid*="TEP1_1" and *VersionId*=3, and also with a reference to "TEP-req-1-1".

This means that a TEP negotiation (booking process) is initiated by a TEPrequest sent from the LSC (TPM) to the LSP. The TEPrequest is identified by the *TEPid*. The *VersionId* need not be used in this case.

All further exchange of TEPs is done by sending TEPresponses (called TEP for simplicity). These TEPs are identified by the *TEPid* and a *VersionId* that is incremented for each exchange. The TEP also has a reference to the original TEPrequest.

In the simplest situation, the LSC sends a TEPrequest with *StatusCode*="Confirmed" to the LSP. Then, the LSP replies with a TEP with *StatusCode*="Confirmed", and the booking is finished. This two-stage confirmation can be used when the LSC and LSP already have established a contract on this service, for instance a blanket contract on an agreed capacity for a period.

In other situations, the LSC needs to see the response from the LSP before he confirms the booking. Then, the LSC sends a TEPrequest with *StatusCode*="NotConfirmed", the LSP replies with a TEP with *StatusCode*="Confirmed", and the booking is finalized by the LSC sending a TEP with *StatusCode*="Confirmed".

During the booking process, the LSC and LSP can use the *StatusReasonCode* to explain the reason why the TEP is set to "NotConfirmed".

A successful booking must always end with two TEP exchanges where both have *StatusCode*="Confirmed".

2.3.4.2. Time Periods in TEP

Figure 10 shows the relationship between the various time periods given in a TEP and how it is used in the story line described in [9].

- *TEP.ServiceStartTimePeriod* and *TEP.ServiceEndTimePeriod*: This is the start and end time for the service(s) described in the TEP, independent of whether this is a transportation service or not (loading, storage etc).
- *TEP.Consignment.PlannedPickupTransportEvent.Period* and *TEP.Consignment.PlannedDeliveryTransportEvent.Period*: This is the time when the goods (consignment) is fetched by the LSP, and delivered to the location defined by the events.
- *TEP.Consignment.MainCarriageShipmentStage.PlannedLoadingTransportEvent.Period* and *TEP.Consignment.MainCarriageShipmentStage.DischargeTransportEvent.Period*: This is the time when the goods (consignment) is loaded by the LSP, and unloaded/discharged at the location defined by the events.
- *TEP.Consignment.MainCarriageShipmentStage.PlannedDepartureTransportEvent.Period* and

TEP.Consignment.MainCarriageShipmentStage.PlannedArrivalTransportEvent: This is the time when the transportation segment starts and ends. In this example, the TEPs describing the transportation only has one shipment stage, meaning that this is the start and the end of the whole transport.

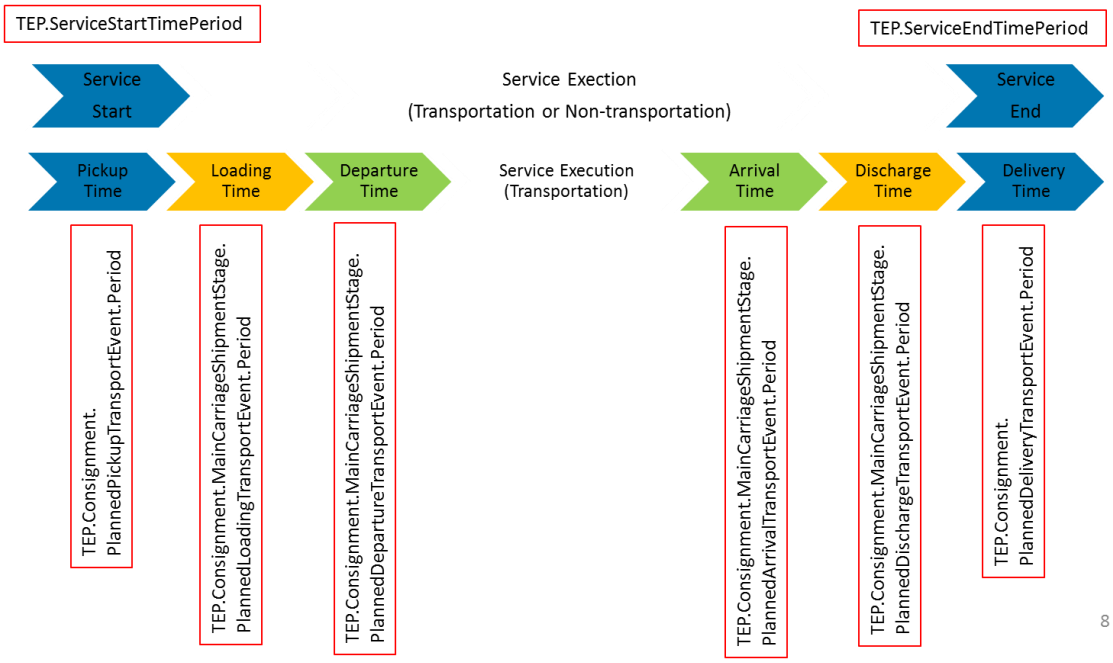

Figure 10 Overview of Time Periods in TEP

2.3.5. Details on TSD

Figure 11 contains the information model for the TSD response, and Figure 12 contains the information model for the TSD request. Classes that are further decomposed are shown in grey. Some of them are listed in Appendix 7.

The TransportServiceDescription class and the TransportserviceDescriptionRequest are the main classes in these structures. The rest of the classes are common structures. During the prototypical work reported in [9], we have kept only one version of the common structures for all five cases (TEP, TEPrequest, TSD, TSDrequest, and TCP).

.

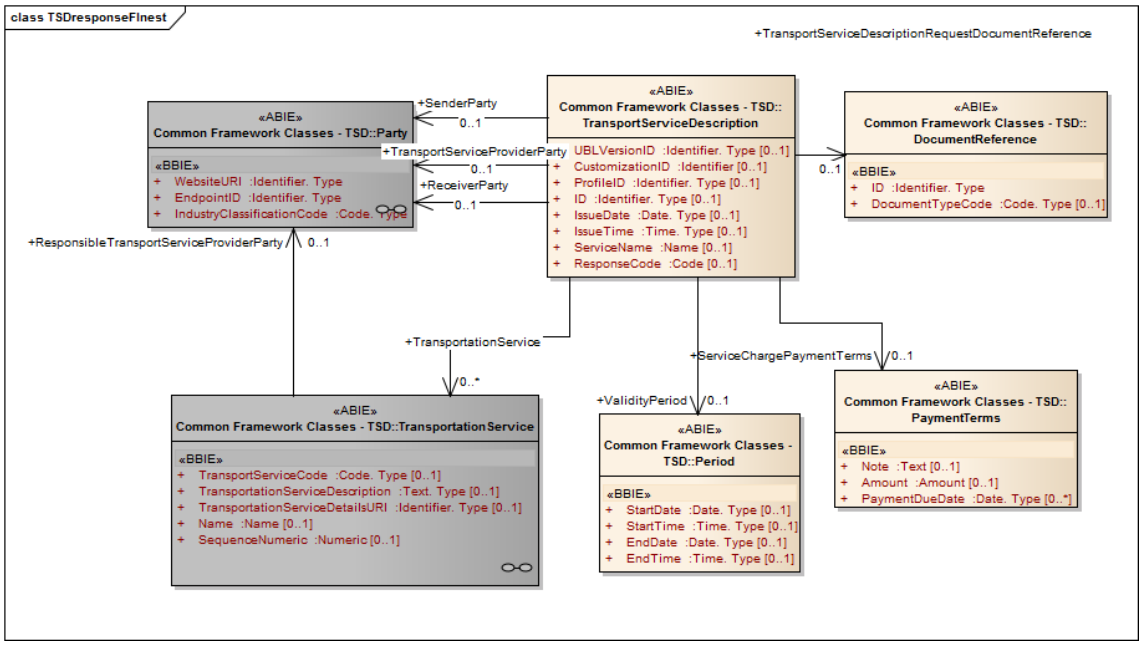

Figure 11 TSD Response

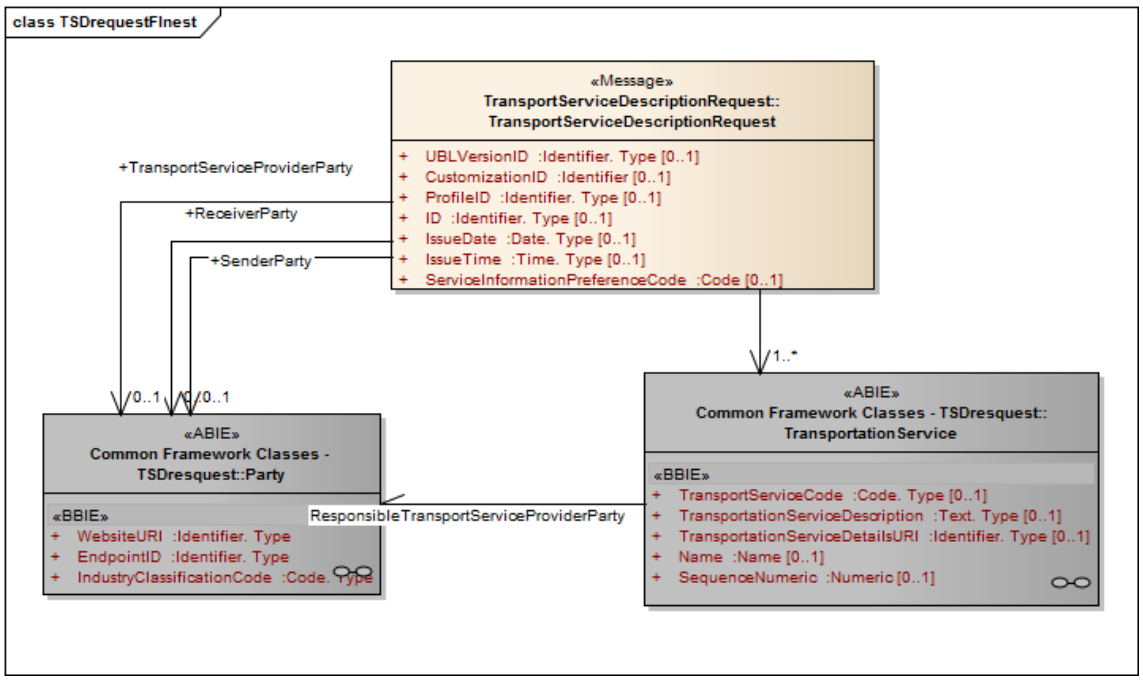

Figure 12 TSD Request

2.3.6. Details on TCP

Figure 13 shows the information model for the TCP – Transport Chain Plan. This data model is based on the GII structure (Goods Item Itinerary) from the UBL 2.1 standard, however, it is simplified compared to D7.3 [8] to cover only what is needed to hold the details of a complete transport chain plan.

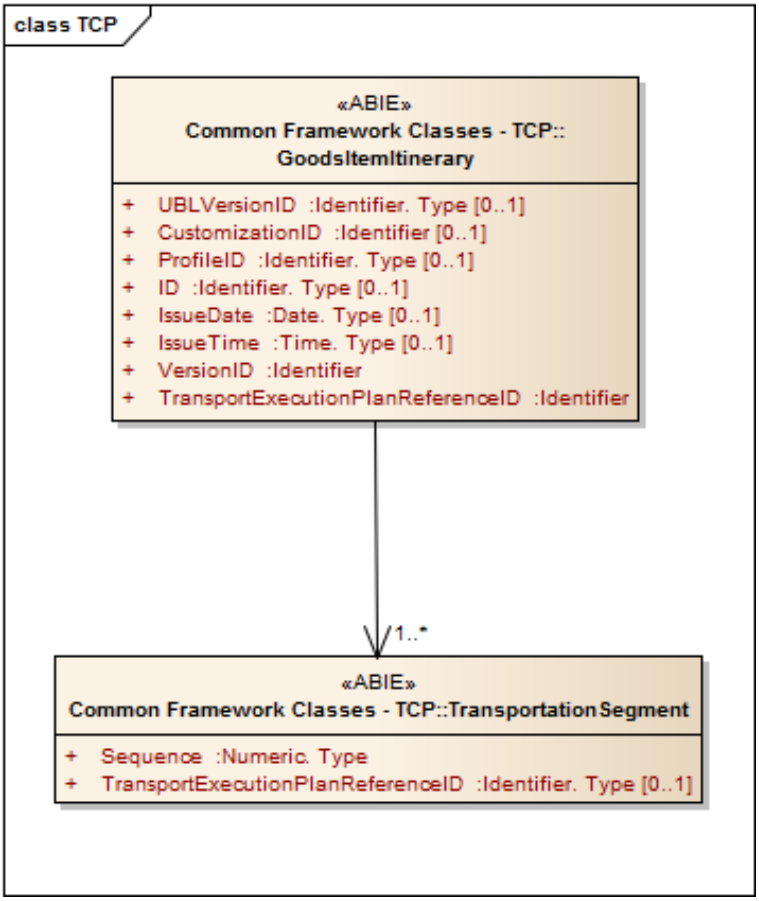

Figure 13 TCP data model

Figure 14 shows an example containing two TCPs. One of the TCPs is identified as TCP1, and this transport chain contains two legs:

1. TEP1.1: A truck transport performed by Sevrin Tranvaag AS
2. TEP1.2: A forwarder service performed by Tyrholm and Farstad

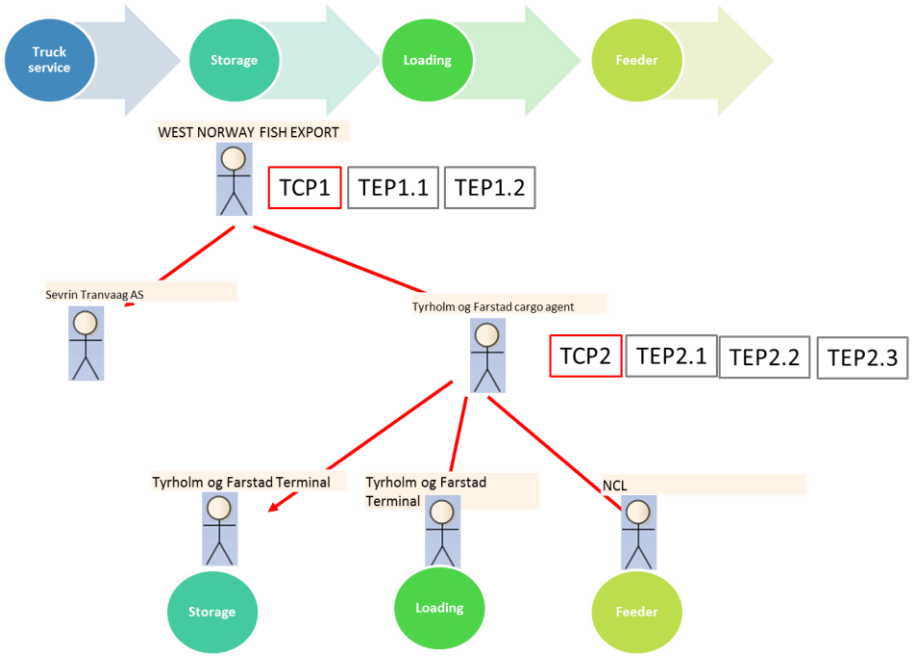

Figure 14 TCP Identification

The other TCP is identified by TCP2 and this transport chain defines the forwarding service consisting of warehousing (TEP2.1), loading (TEP2.2) and feedering (TEP2.3).

From Figure 13, we have the following usage of identifications:

1) *GoodsItemItinerary.ID*: Identification of the TCP, that is, of the transport chain plan. In this example, either TCP1 or TCP2.
2) *GoodsItemItinerary.TransportExecutionPlanReferenceID*: Identification of the TEP that is part of the transport chain described by this TCP. In our example, this value is blank for TCP1. For TCP2, the value is TEP1.2, meaning that it holds the reference from TCP2 to TEP1.2.
3) *TransportationSegment.TransportExecutionPlanReferenceID*: This is the reference to the TEPs that is contained in a certain TCP. In our example, TCP1 contains two transportation segments, namely TEP1.1 and TEP1.2, and TCP2 contains three transportation segments, namely TEP2.1, TEP2.2 and TEP2.3.

We also need a link from each TEP to the TCP that it is contained in. This is done by setting the element *TEP.AdditionalDocumentreference.ID* equal to the *GoodsItemItinerary.ID*, and by setting *TEP.AdditionalDocumentreference.DocumentTypeCode* equal to 441, meaning "Transport Routing Information". Figure 15 shows how this may look like for TEP1.1 and TEP1.2 in the example.

```
<cac:AdditionalDocumentReference>
    <!-- Reference to the TCP that this TEP is contained in -->
    <cbc:ID>TCP1</cbc:ID>
    <!--441= transport routing information. -->
    <cbc:DocumentTypeCode>441</cbc:DocumentTypeCode>
</cac:AdditionalDocumentReference>
```

Figure 15 TCP Identification Example for TEP1.1 and TEP1.2

2.4. Changes to the FInest Core Model

The following changes have been done to the FInest Core model compared to the eFreight core model, Table 10. "Fetched from UBL2.1" means that the attributes are not found in the eFreight core, but that they are found in the UBL2.1 and added from there.

"Extention of UBL2.1" means that this is not found in the UBL2.1 model, but that it is added due to FInest needs. These changes means that the FInest model remains compliant with the UBL2.1 standard.

The changes for the TEPresponse summarized in Table 10 are also valid for the TEPrequest. The only difference is that the planned transport events are removed from the ShipmentStage for the TEPrequest.

Main Class	Class	Changes Done	Relations to UBL/eFreight
Common	Party	Added attributes: EndPointID WebsiteURI IndustryClassificationCode	Fetched from UBL2.1
Common	DocumentReference	DocumentTypeCode	Fetched from UBL2.1
Common	TransportExecutionTerms	Removed ChangeConditions	
Common	TransportationService	Added TransportationServiceDetailsURI	Fetched from UBL2.1
Common	Location	Removed LocationTypeCode Removed InformationURI	Removed compared to eFreight core
Common	Consignment	Removed CarrierAssignedID	Removed compared to eFreight core
Common	Consignment	Removed	Removed compared to

Main Class	Class	Changes Done	Relations to UBL/eFreight
		OriginalDepartureCountry FinalDestinationCountry	eFreight core
Common	Consignment	Removed PreCarriageShipmentStage OnCarriageShipmentStage	Removed compared to eFreight core
Common	TransportHandlingUnit	Removed TotalGoodsItemQuantity ShippingMarks	Removed compared to eFreight core
Common	ShipmentStage	Added ArrivalCustomsTransportEvent DepartureCustomsTransportEvent	Fetched from UBL2.1
Common	ShipmentStage	Added PlannedDischargeTransportEvent PlannedLoadingTransportEvent RequestedDischargeTransportEvent RequestedLoadingTransportEvent	Extension of UBL2.1
Common	ShipmentStage	Removed TransportMeans	Removed compared to eFreight core
Common	TransportEvent	Added Contact	Added from UBL2.1
Common	GoodsItem, Dimension, Item	Kept as described in UBL2.1	
Common	TransportEquipment	Removed TransportEquipmentSeal	

Table 10 FInest Core Model Adaptions for TEPresponse

Adaptions for FInest regarding TSDresponse is summarized in Table 11.

Main Class	Class	Changes Done	Relations to UBL/eFreight
TSDresponse	TSDresponse	Added attributes: ServiceName	Fetched from UBL2.1

Main Class	Class	Changes Done	Relations to UBL/eFreight
		ResponseCode	
Common	TransportationService	Added TransportEquipment	Fetched from UBL2.1
Common	TransportationService	CommodityClassification	Fetched from UBL2.1
Common	TransportationService	Period	Fetched from UBL2.1
Common	TransportationService	EnvironmentalEmission	Fetched from UBL2.1

Table 11 FInest Core Model Adaptions for TSDresponse

The TSDrequest in FInest has the structure as shown in Figure 12, and the same adaptions as described in Table 11 are done.

2.5. Transport Planning Module Interaction

Figure 16 shows how the TPM interacts with the other FInest core modules and the Backend and Frontend during transport planning.

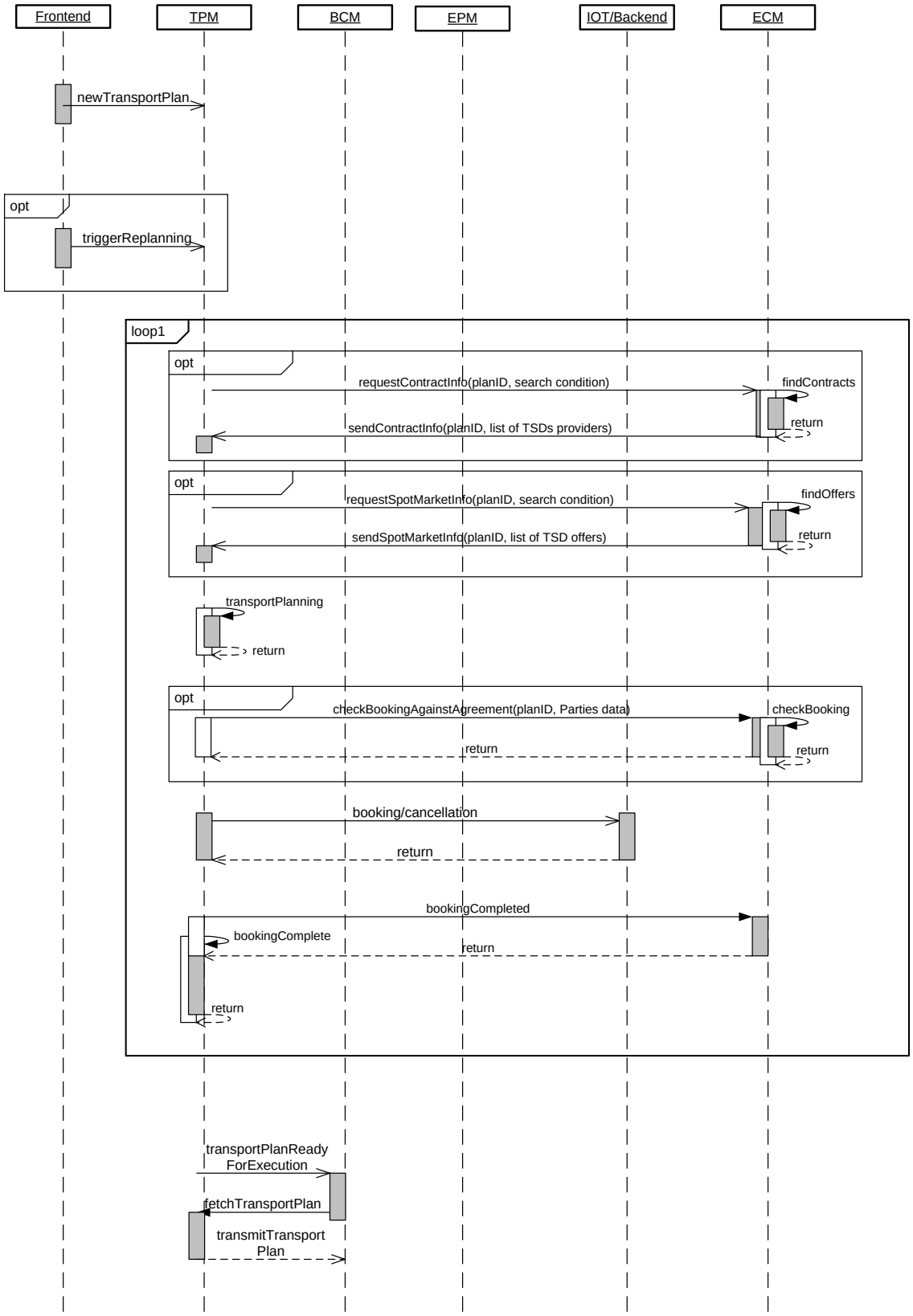
Frontend
TPM
BCM
EPM
IOT/Backend
ECM
newTransportPlan
opt
triggerReplanning
loop1
opt
requestContractInfo(planID, search condition)
findContracts
return
sendContractInfo(planID, list of TSDs providers)
opt
requestSpotMarketInfo(planID, search condition)
findOffers
return
sendSpotMarketInfo(planID, list of TSD offers)
transportPlanning
return
opt
checkBookingAgainstAgreement(planID, Parties data)
checkBooking
return
return
booking/cancellation
return
bookingCompleted
bookingComplete
return
return
transportPlanReady
ForExecution
fetchTransportPlan
transmitTransport
Plan

Figure 16 Sequence Diagram of TPM interactions during Planning

Figure 17 shows how the TPM interacts with the other core modules during replanning. The Request for replanning is triggered by the user, as a result of certain notifications received regarding a detected situations. Replanning is perceived by the TPM as the creation of a new transport plan, and the planning process is repeated from the point where the failure occurred.

Replanning can be triggered in two different ways, user initiated replanning or event initiated replanning. User initiated replanning means that the TPM is notified directly by the front end that the user wants to start replanning. Event initiated replanning is initiated from Backend (events received). and the BCM sends a replanning trigger to the TPM containing execution status information on the actual TCP and the reason why a replanning is needed.

After the initialization, a regular planning process is started with the original demand and the information received from the BCM available.

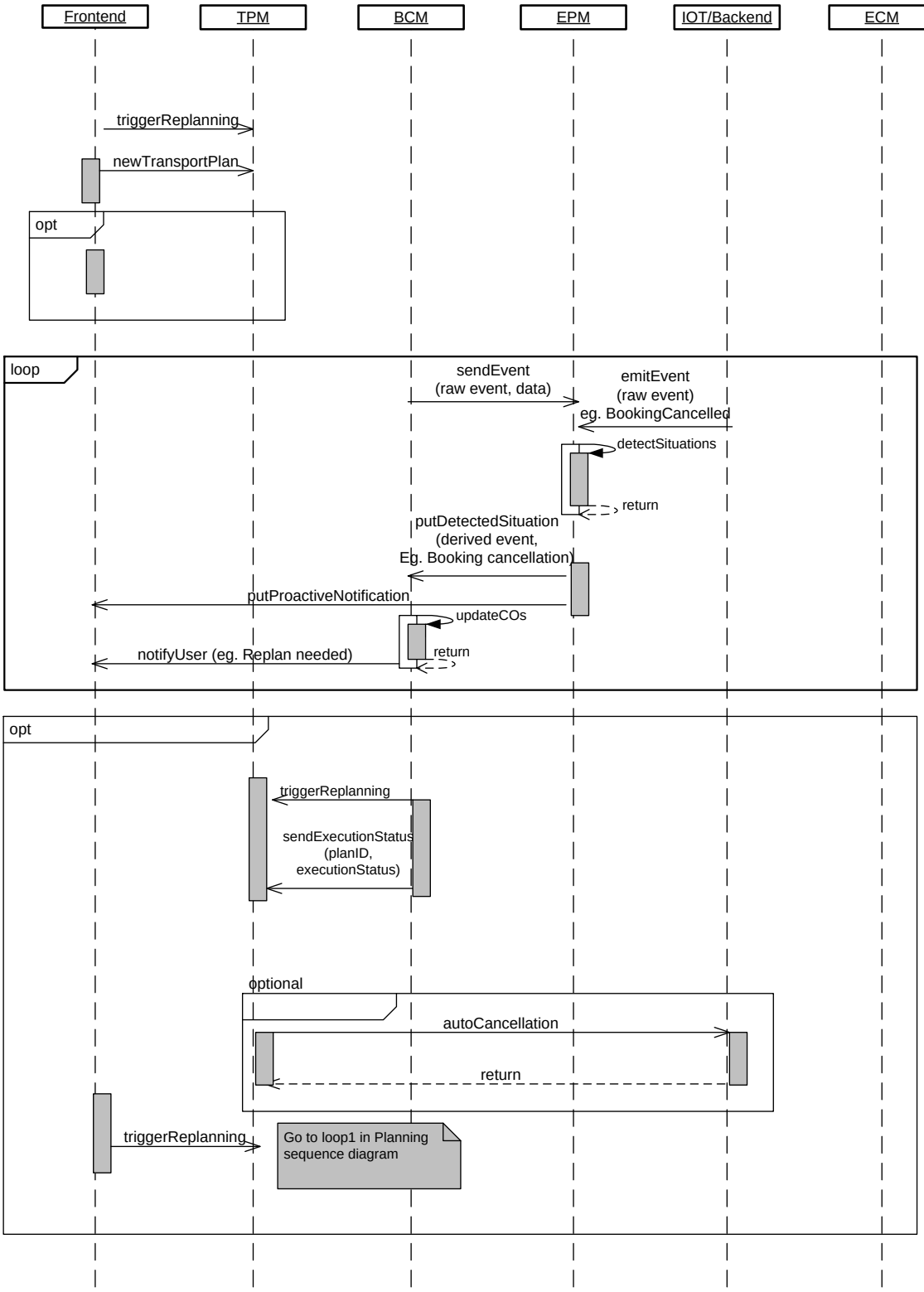

Figure 17 Sequence Diagram of TPM Interactions during Replanning

2.6. Transport Planning Module User Role Description

This section describes the roles of the users of the TPM during the transport planning process. For each role, a set of access rights (create, read, update, delete), information elements (TCP, TEP, TSD), and access handling rights (transfer ownership, add and remove others from a specific role) are specified. It should be noted that these are requirements to the framework system (e.g. the FInest platform) that are used is access handling, and are not implemented in the TPM itself.

The roles of the users of the TPM are summarized in the table below. More description of the roles related to the story line is found in [11].

Table 12 - User roles for TPM

Role in planning	Information Elements	Access	Access handling rights
Owner	All	CRUD	Give all kinds of accesses to others Remove contributor and reader rights for others. Remove owner rights for self.
Contributor	Description contributor: Access to demands (for LSCs) or services (for LSPs)	RU	None
	TEP contributor: Access to TCP and TEP		None
Reader	Description reader: Access to demands (for LSCs) or services (for LSPs)	R	None
	TEP reader: Access to TCP and TEP		
Negotiator	Access to TEP Right to enter negotiation	CRUD	Give read access to TEPs

2.6.1. The Owner Role

The owner role has unlimited access to all the owner's actual information elements, be it TCP, TEP, demand description, or service descriptions (TSDs). To summarize, an owner as the following access to TPM information elements:

- LSCs has CRUD access to demands
- LSPs has CRUD access to services (TSDs)
- LSCs may initiate search for transport services for demand where he is the owner
- LSPs may initiate search for demands for demands where he is the owner
- LSCs may put the demand on a marketplace
- LSP may put services (TSDs) on a marketplace
- Both LSCs and LSPs have CRUD access to TEPs that are part of the plan
- The LSC has CRUD access to the TCP describing the whole plan.

In addition, the owner has the right to forward the ownership of the information elements to other users. This means that the current owner sets another user as the new owner of the information. If there are more than one owner, the owner has the right to remove itself from the owner list (in this way, ownership of a plan can be transferred to another user). To summarize, the owner role as the following access control administration rights:

- May add others to owner role for information elements he has owner rights to.
- May add and remove others to the other roles for information elements he has owner rights to.
- May remove himself as owner, given that at least one other owner exists[1]. He cannot change the ownership of finalized TEPs, since the owner of TEP is always (at least) the two involved parties (the LSC and the LSP).

2.6.2. The Contributor Role

The role Contributor is used to restrict the access to only Read and Update access to the information elements. The information elements is related to either demand or service descriptions, or to transport plans. Note that a single actor can be given several roles, i.e. an actor can have full contributor rights through the whole planning process by having both description and TEP contribution roles.

The Description Contributor Role

This role has read and update access to demand descriptions or to service descriptions. This role is typically used after the transport plan is created and it is used to control the access to the related demand description and service description (TSD). Typically, a LSC will have access to the demand descriptions, and a LSP will have access to the service description (TSD).

The TEP Contributor Role

This role has read and update access to TCPs and to TEPs, that is, to the whole transport plan and to separate legs. This role is typically used by the LSC to manage the whole transport plan (TCP), or a single TEP. It can be used by a LSP to manage information in a TEP.

[1] An example of this is found in mockup 4 (D7.2) Here, the initial data for the demo is set up by the Korean exporter (the factory), but the party that should be "plan owner" is Arcelik. The plan is initially owned by the exporter (as it is the exporter that initiates the plan), but as soon as possible, Arcelik is set up as an owner, and the exporter removes himself from the plan owner list (while still being a contributor).

2.6.3. The Reader Role

The reader role only has read access to information elements in the TPM. As for the contributor role, this role is of two types:

The Description Reader Role

The Description Reader role has read access to a demand description and to a service description (TSD). A LSC will have access to the demand, and a LSP will have access to the service description.

The TEP Reader Role

The TEP Reader role has read access to the transport plan information, that is, to the TCP describing the whole plan, or to a TEP description one agreement. Both LSCs and LSPs are involved here.

2.6.4. The Negotiator Role

A separate role covering the possibility to do booking negotiation is needed in the TPM. This role has Create, Read, Update, and Delete access to selected TEPs, and has the right to do negotiation (sending and responding to TEPrequests and TEPresponses). This role has the possibility to give read access to other users to selected TEPs.

3. Generic Enabler Usage

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 SQL/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. However, several GEs will be used through the FInest platform, like GEs for security and connections to legacy systems.

4. Plans for Phase II

As this is the final deliverable from the transport planning and replanning work package, there will not be any further work with the TPM within the FInest project. The work on the Transport Planning Module will however be continued in the second phase of the FI-PPP-Program as a part of the cSpace project.

In the cSpace project, the continuation of the work on the TPM will be part of the work on "cross-domain baseline applications", where the TPM will be the base of the *"Logistics planning app"*.

4.1. Description of the cSpace Logistics Planning app

The following description of the cSpace "Logistics Planning App" was used in the cSpace proposal:

"The aim of this baseline app is to provide logistics planning functionality both for the logistics service clients and providers. The app is based on the Transport Planning Module from the FInest project."

For the logistics service client, the baseline app shall support the building of a transport chain plan based on the client's demand and online available services, using the latest available information for service descriptions. For the provider, the baseline app will enable the description of transport services that can be used by the clients, enable the use of markets to find demands that match the provided services, as well as for planning the use of subcontractors. For both client and provider, the baseline app will have facilities for detailing out the execution plans and for negotiation and booking of the services. The app will also provide functionality for replanning, in case the execution of the original plans fails.

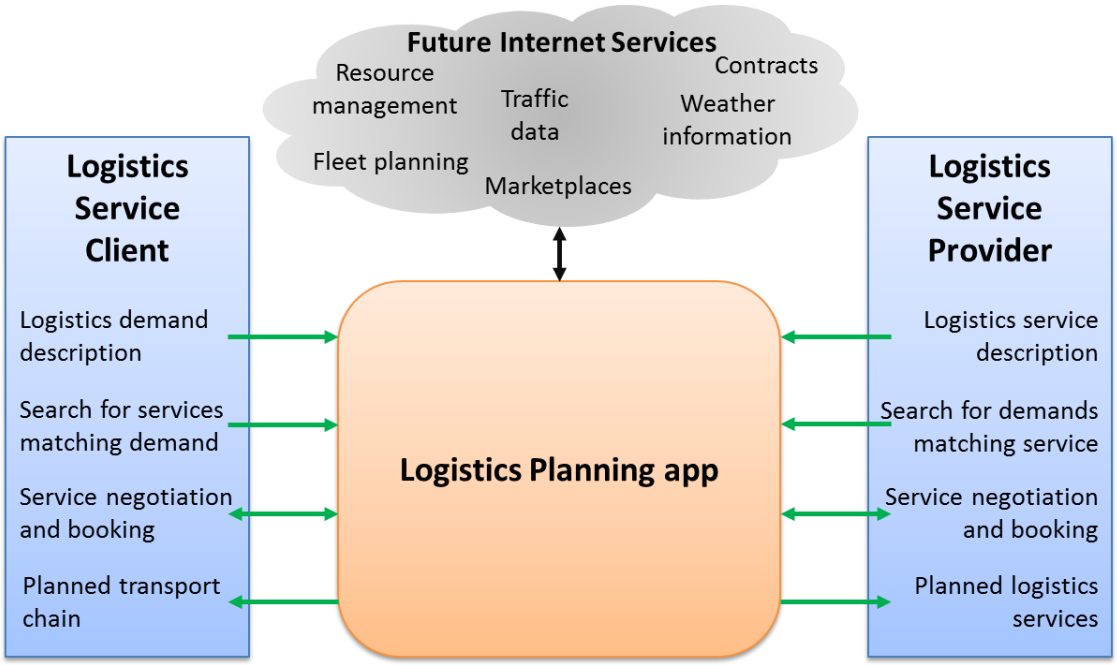

Figure 18 cSpace Logistics Planning app

4.2. Development in phase 2

Given the description above, the main work for the app in cSpace project will be the implementation of the app. Also, the work needs to be aligned with the work on the cSpace core (i.e. platform level functionality) as well as the work on the cSpace equivalent of the FInest ECM, as this will be the main contributor of contract and marketplace information that is used by the app.

In the current stage, the overall design of the TPM cover the main features of the Logistics Planning app, and will probably be used (with possible minor modifications) in the cSpace work. The implementation of the system in cSpace may reveal further needs for creating a more detailed design for some of the components, or a need for further clarifications of interfaces and functionality. It is assumed that the design will be updated accordingly during the cSpace project.

The development of the logistics planning app should be open for minor additions or modifications to functionality and interfaces if such modifications could be considered a natural part of the app. When updating the functionality, it should however always be kept in mind that functionality changes often can be achieved by modifying the user interface (e.g. tailoring the UI to specific user needs) or using external systems (e.g. add-ons that utilizes the already defined interfaces and functionality) without changing the specifications of the app itself. Any updates to interfaces or functionality of the app must of course be properly documented.

Implementation-wise, the prototype (delivered as D7.4 of the FInest project) is on a level where it can be used for concept demonstration purposes, but is in no way close to a finished product; thus most development effort on the cSpace Logistics Planning app should be on implementing the system. While some of the components already implemented in FInest may be useful for the cSpace app, the developers of the app may also prefer to re-implement the components from scratch.

4.3. Development Plan for Phase 2

4.3.1. Task Description

The tasks for development of the Logistics Planning app will follow the same structure as for the other baseline apps in the cSpace project, and will consist of the following tasks:

- Implementation of the Logistics Planning app based on both the cSpace functionalities and the Generic Enablers provided by the Core Platform
- Definition of a testing plan for the application
- Testing the application based on the testing plan
- Management of the iterative cycles of development of the application
- Complete the technical documentation of the application, which will be included in a technical wiki

4.3.2. Deliverable Description

The following deliverables are relevant for the development of the Logistics Planning app, Table 13:

No	Deliverable Name	Type	Delivery Month	Description
D400.6	Functionalities of baseline apps	Report	3	For the Logistics Planning app, this will be a recap of the current (i.e. FInest TPM). If necessary, the deliverable may also contain changes needed to fit the application to the cSpace platform and test cases.
D400.7	Baseline applications 1st release	Prototype	9	This includes a prototype release from the first development iteration of the Logistics Planning app
D400.8	Baseline applications 2nd release	Prototype	15	This includes a prototype release from the second development iteration of the Logistics Planning app
D400.9	Baseline applications 3rd release	Prototype	21	This includes a prototype release from the third (and final) development iteration of the Logistics Planning app

Table 13 List of Relevant Deliverables in cSpace Project for Logistics Planning app

The planned timeline will be as follows:

Review of the functionality of the TPM for use in the cSpace Logistics Planning app will take place in months 0-3 of the project. This may include minor changes to functionality to better fit the scope of cSpace, as well as clarifications of functionality descriptions if this is necessary. The main work on functionality descriptions will be completed for the cSpace D400.6 deliverable.

The actual development (implementation and design updates) will take place in months 3-21 of the project. There will be three main releases of the Logistics Planning application, at months 9, 15 and 21. During the whole work with the application, documentation must be updated to reflect the actual updates to functionality and interfaces, as well as the technical structure of the application.

The following Gantt diagram shows the proposed timeline and milestones (black lines at month 3, 9, 15, and 21) for the development of the Logistics Planning app:

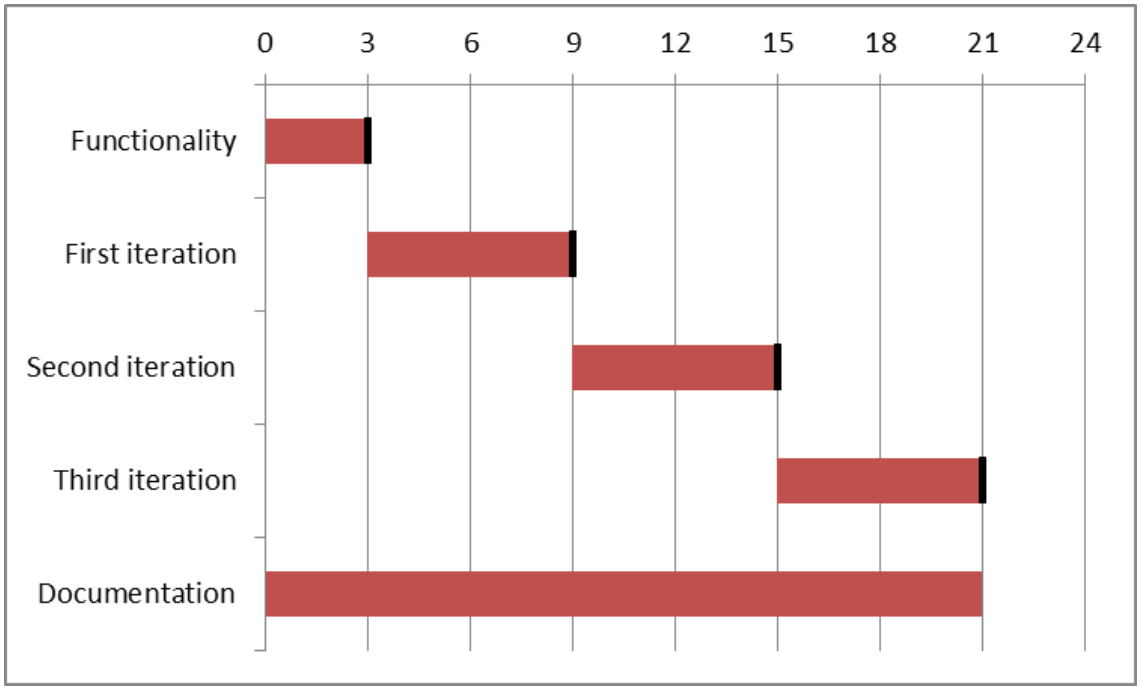

Figure 19 Gantt chart for development of Logistics Planning app

5. Summary and Discussion

This deliverable describes the final version of the technical specification of the Transport Planning Module in the Finest project. This consists of a proposed architecture for the module as well as interface and information model specifications. The module is based on concepts and requirements from earlier deliverables from the Transport Planning and Replanning Work Package in the project, and the design described in D7.3, with additions and changes done during prototype implementation of the system.

While this is the last deliverable for the Transport Planning and Replanning work in the FInest project, the work will be continued within the Logistics Planning application of the cSpace as the follow-up project of FInest. It is believed that the design presented in this deliverable will work as a valuable base for the work that is to be performed in the cSpace project.

During the work with the TPM, a prototype system has been implemented. While the prototype serves to demonstrate some of the main intended functionality of the TPM, it is still far from a complete system, thus large parts of the design presented in this deliverable has not been tested in practice, and it may still be a need to detail, clarify or change parts of the system design during the development of the Logistics Planning application in cSpace. We do, however, believe that the overall design structure can be used without major modifications.

The module should also be open for changes that enhances or adds functionality, but it is our belief that additional functionality can also be achieved by either designing "add-on modules" or by creating specialized user interfaces for a given use of the module. Such add-ons or specialized interface can utilize the already described interfaces and functionality, thus creating

added functionality, specialization or enhancements to the system without changing the original TPM structure.

References

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

[2] "eFreight Project Web Site," [Online]. Available: http://www.efreightproject.eu/.

[3] "FreightWise project website," [Online]. Available: http://freightwise.info/.

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

[5] "DiSCwise project homepage," [Online]. Available: http://www.discwise.eu/.

[6] 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.

[7] 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.

[8] M. Hagaseth, Å. Tjora, K. E. Fjørtoft, L. Ramstad and C. Steinebach, "FInest D7.3 - Initial Technical Specification of the Transport Planning Component," 2012.

[9] M. Hagaseth, C. Steinebach, Å. Tjora, L. Ramstad and A. Rialland, "FInest D7.4 - Prototypical Implementation of the Transport Planning Component," 2013.

[10] [Online]. Available: http://en.wikipedia.org/wiki/K_shortest_path_routing 01.03.13.

[11] M. Hagaseth and C. Steinebach, "D7.4 Prototypical Implementation of the Transport Planning Component," 2013.

[12] C. C. Marquezan, S. Heyne, M. Stollberg, N. Golmohamadi, A. Metzger, M. Hagaseth, Å. Tjora, C. Alias, Ø. Olsen, M. Zahlmann and M. Turkay, "FInest D8.5 - Final Technical Specification and Phase II Implementation Plan for Logistics Contract Manager," 2013.

[13] "UBL 2.1," [Online]. Available: http://docs.oasis-open.org/ubl/prd2-UBL-2.1/mod/summary/reports/UBL-AllDocuments-2.1.html.

[14] "Common Framework," [Online]. Available: http://www.its.sintef9013.com/CF/v01/. [Accessed 08 03 2013].

6. Appendix A: TEP Information Model Details

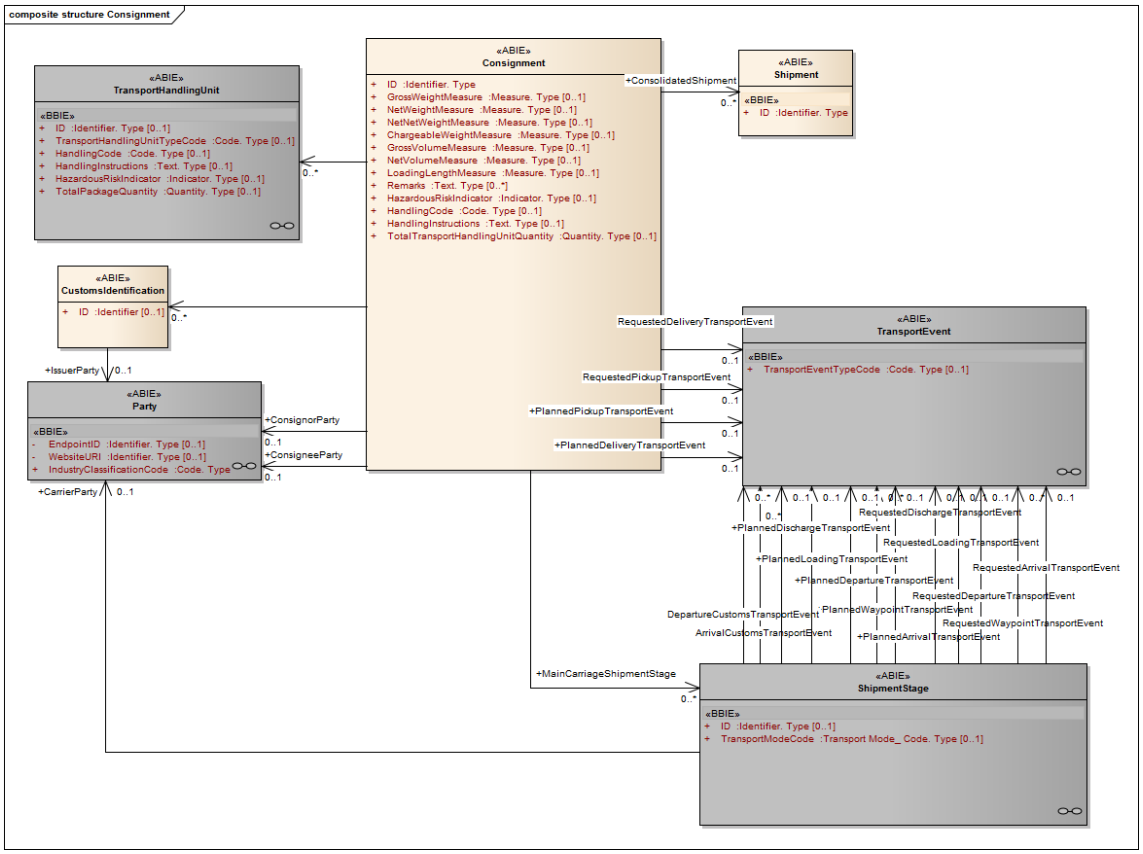

Figure 20 Consignment

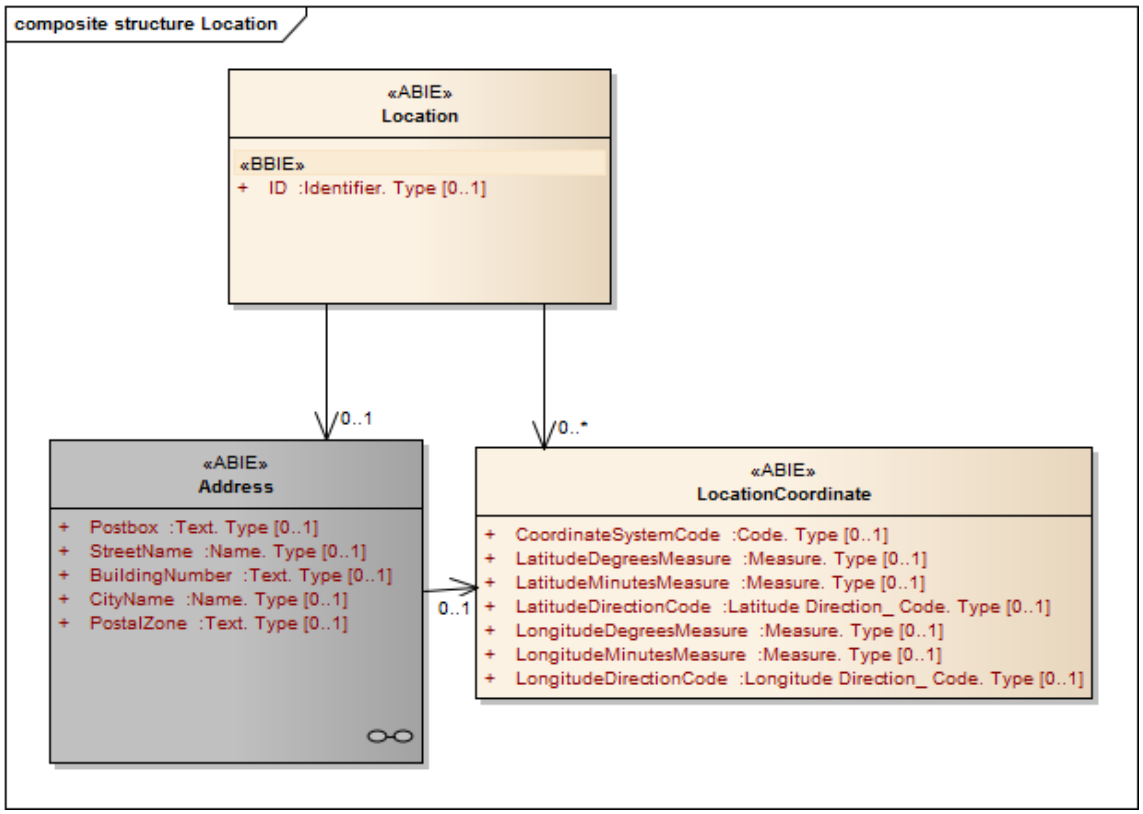

Figure 21 Location

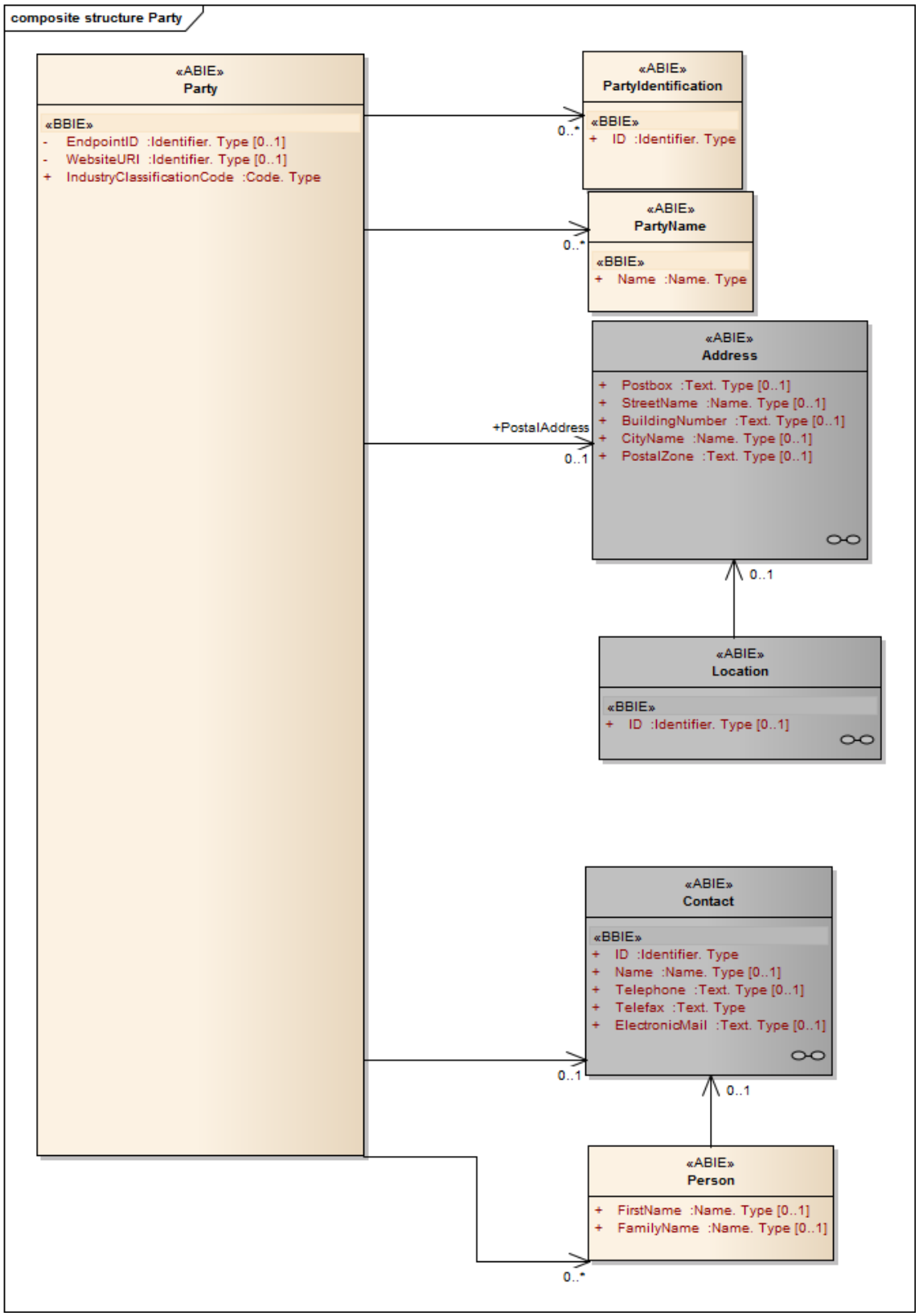

Figure 22 Party

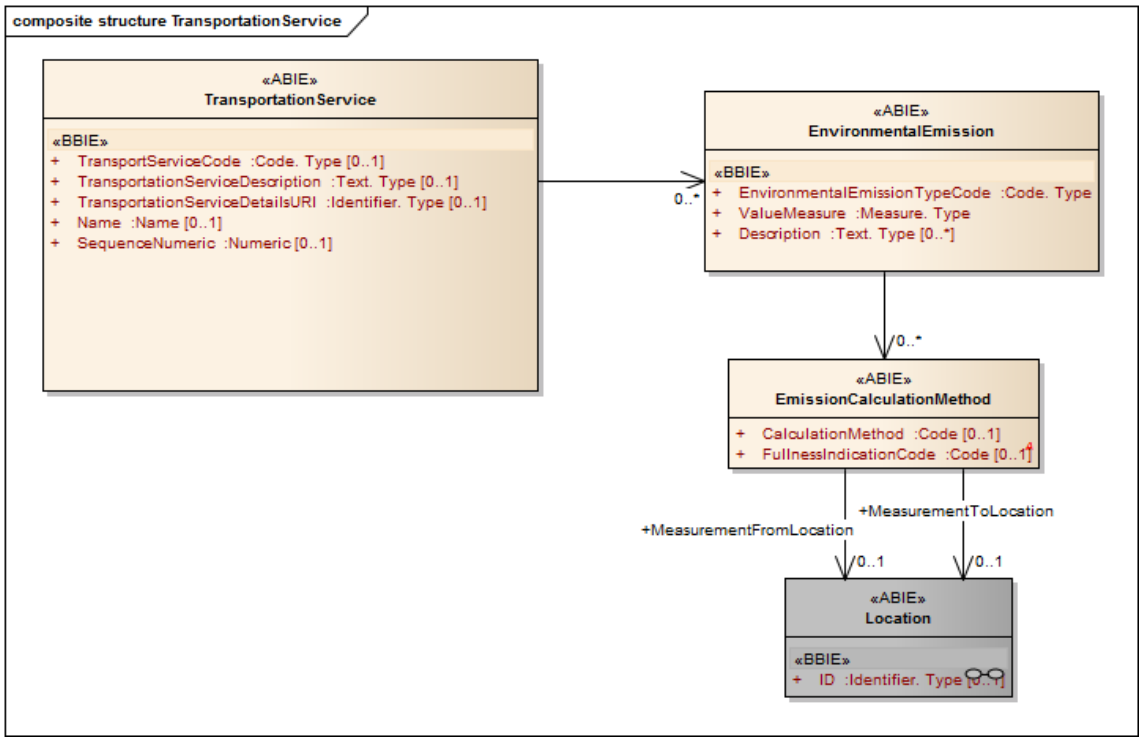

Figure 23 Transportation Service

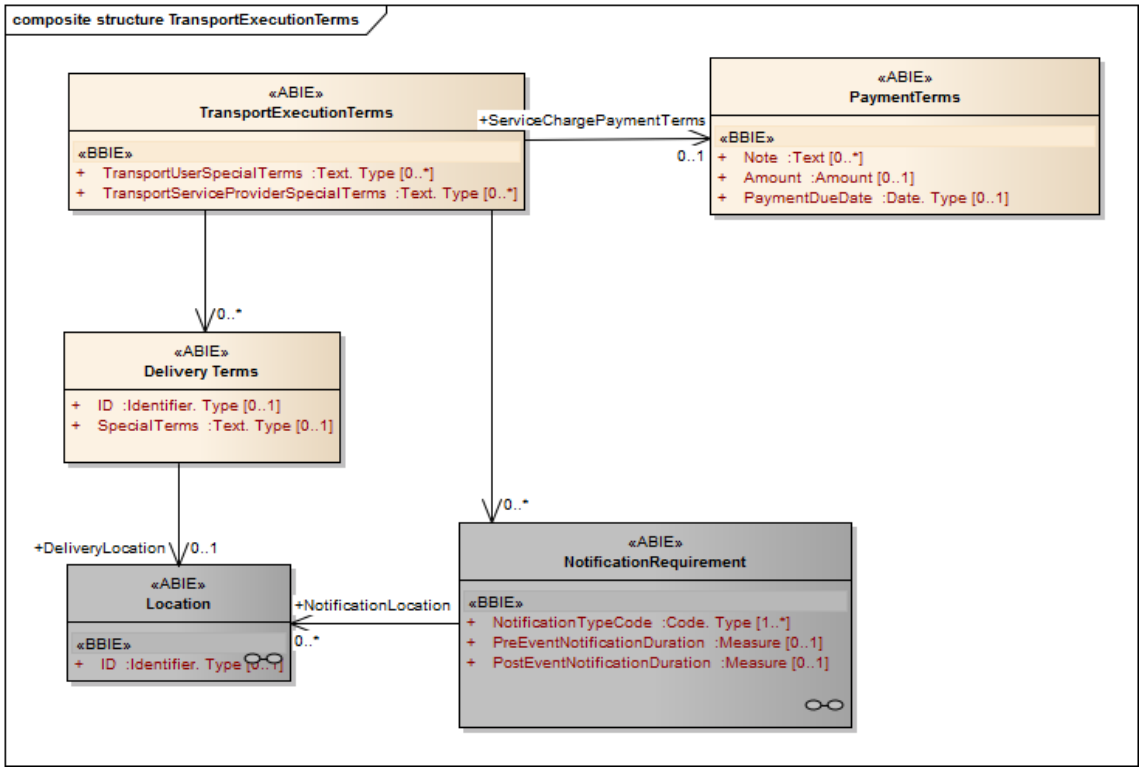

Figure 24 Transport Execution Terms

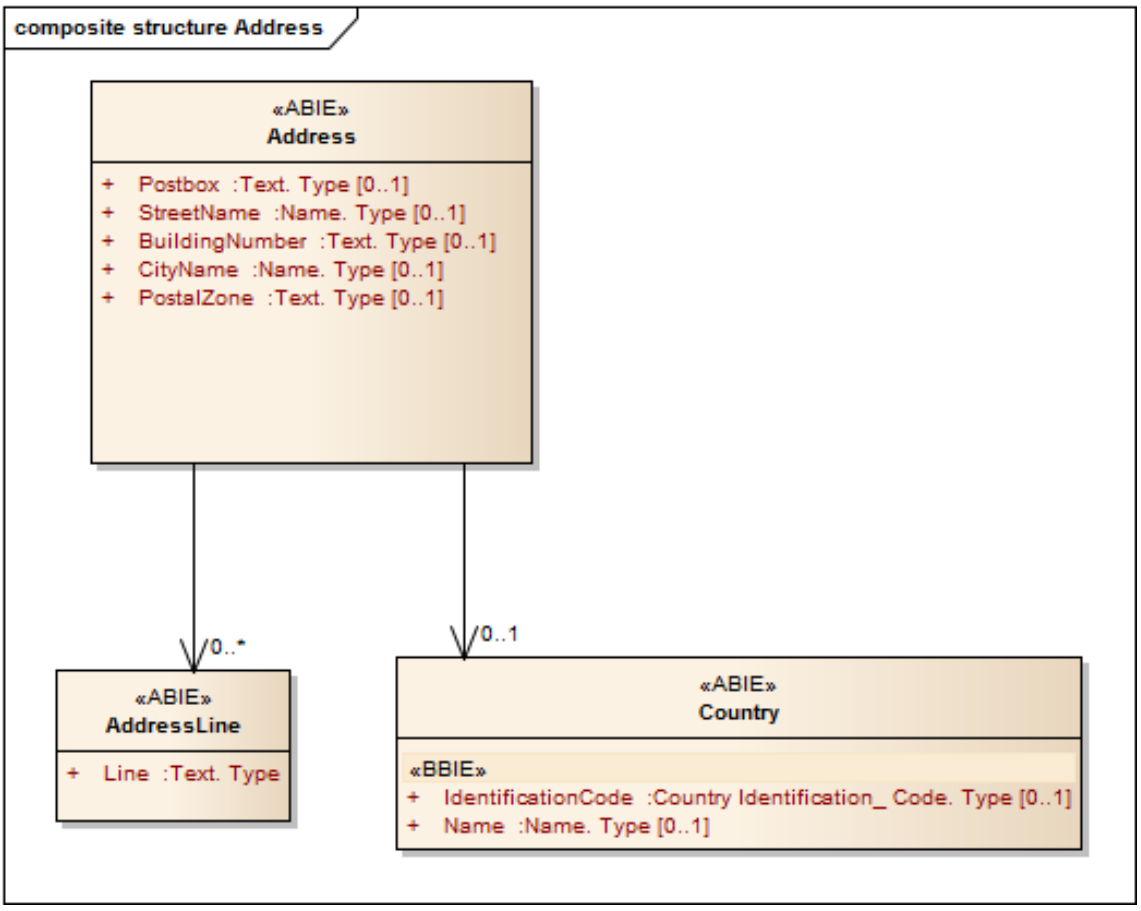

Figure 25 Address

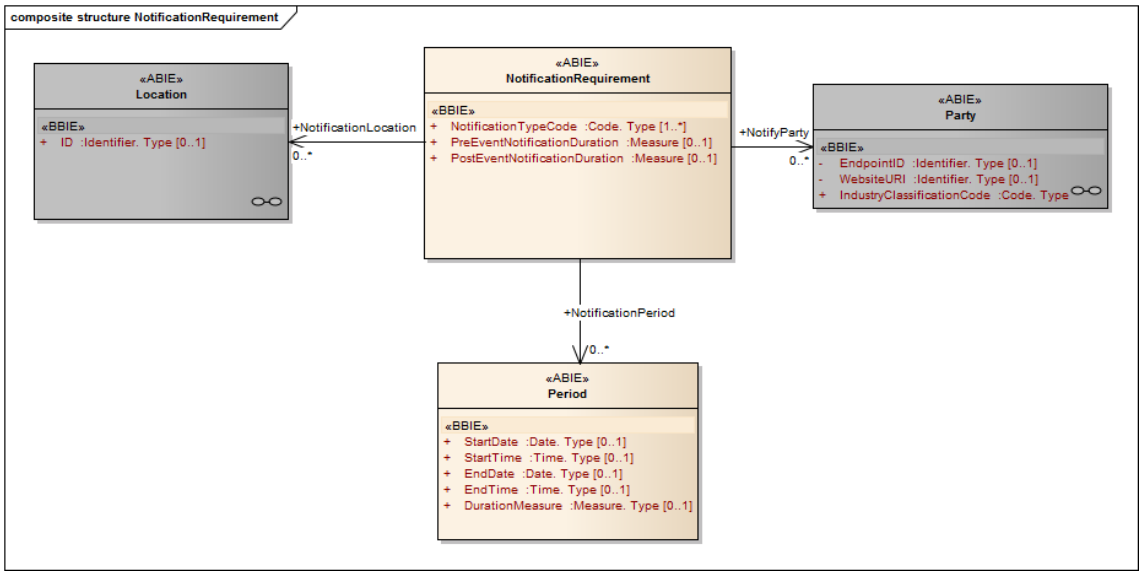

Figure 26 Notification Requirements

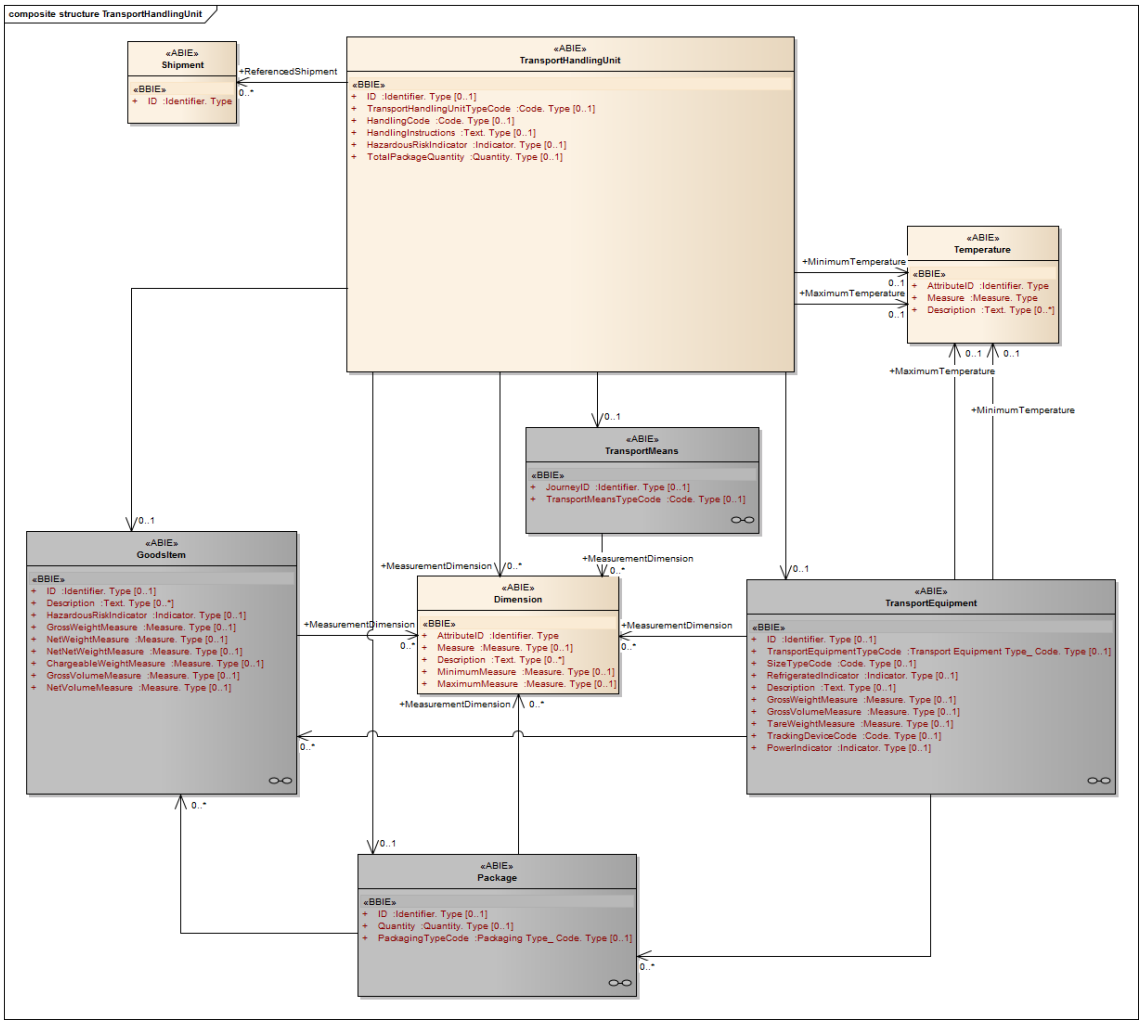

Figure 27 TransportHandlingUnit

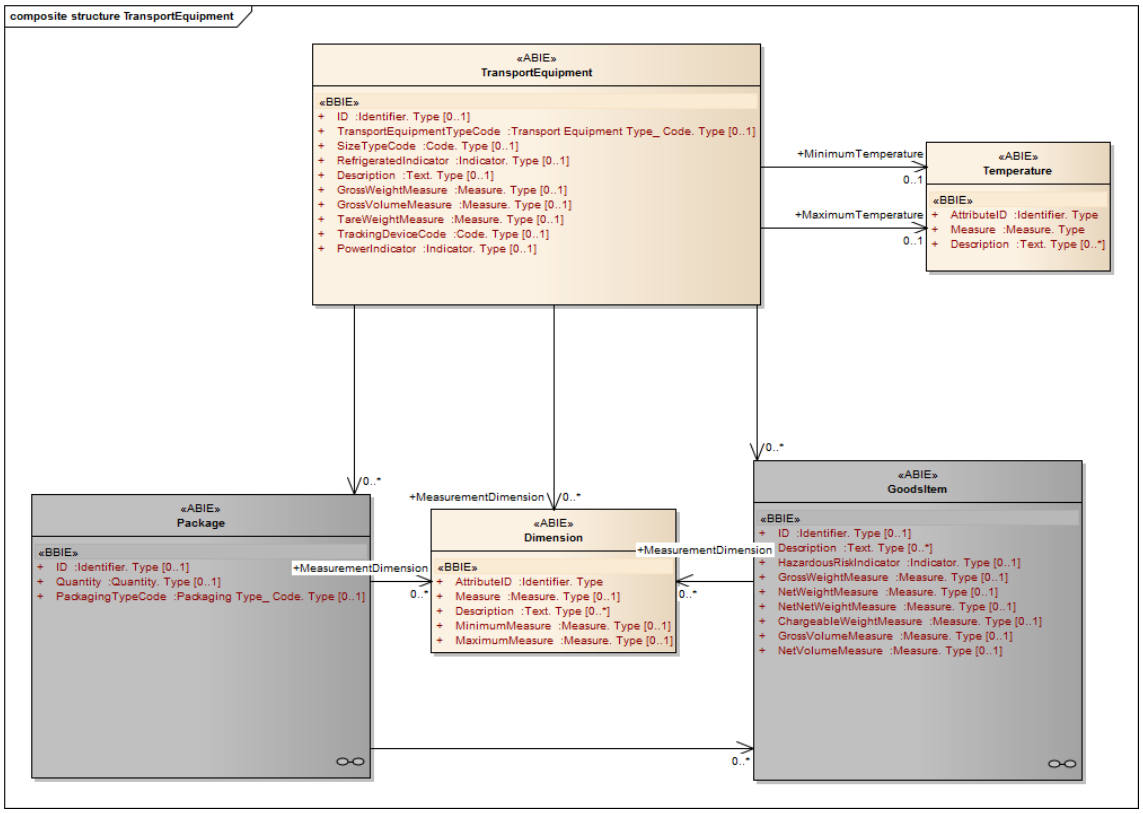

Figure 28 TransportEquipment

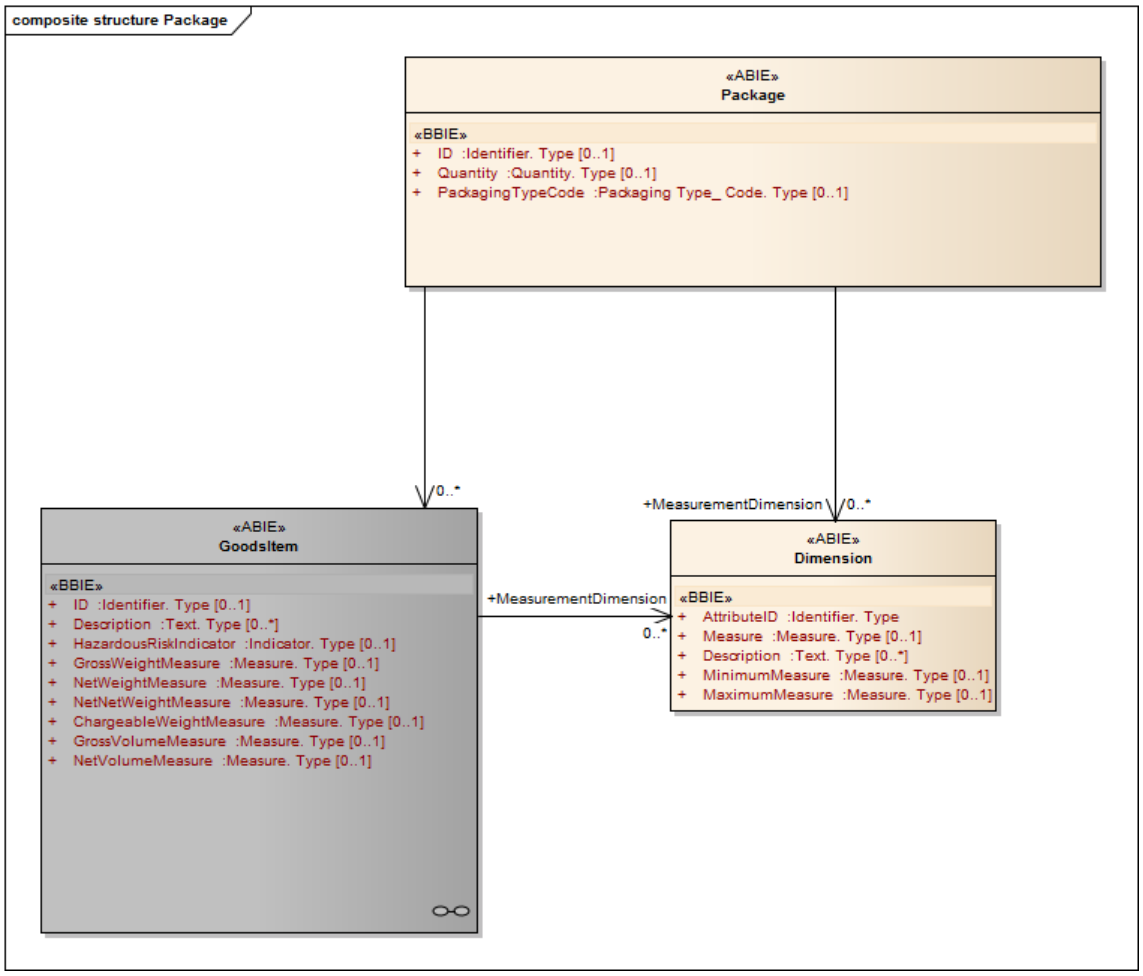

Figure 29 Package

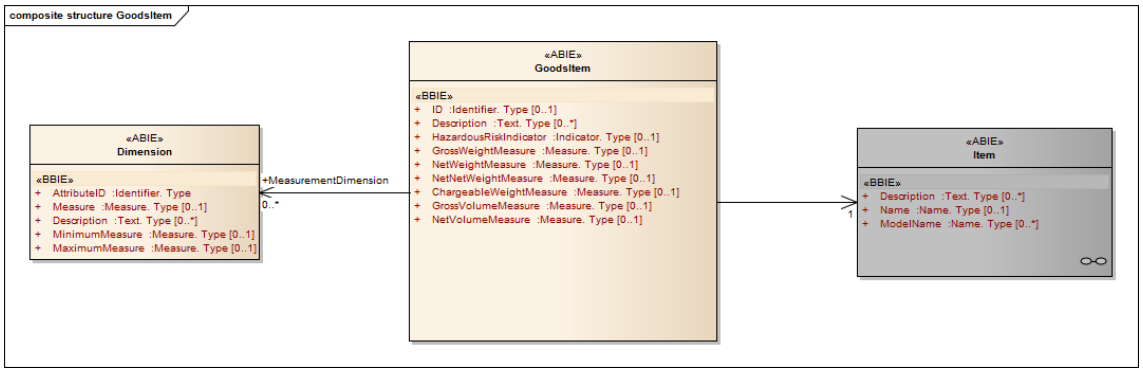

Figure 30 GoodsItem

7. Appendix B: TSD Information Model Details

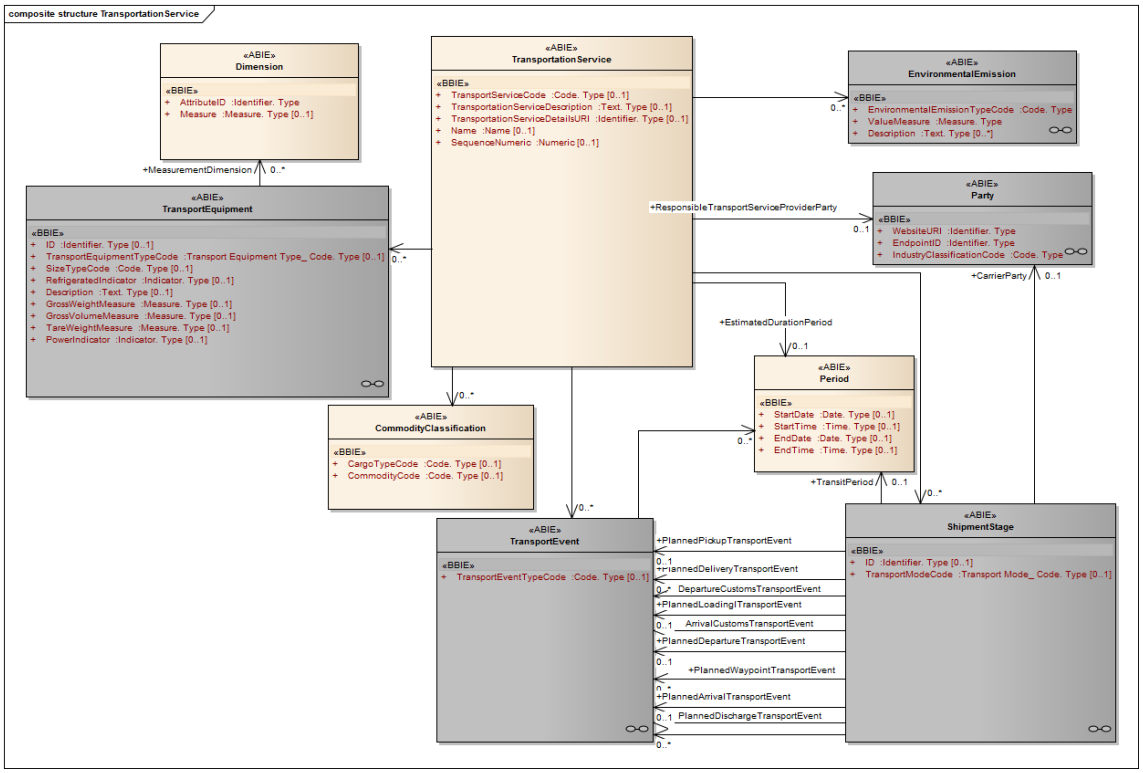

Figure 31 Transportation Service for TSD response